FUJITSU PERSONAL COMPUTER MAGAZINE
パソコン情報誌

FUJITSU MICRO 7 8 11

'83 第3号 隔月刊

480円

マシン語特集

三次元パッケージのアルゴリズム

FM-8の命令を増やそう!

ゲームリスト(B-1核爆撃機/アルデバラン)

信頼と創造の富士通

FM-7 MAZE

¥126,000（本体価格 簡易言語ソフト付）

FM-7

セブン

興奮パソコン。

新発売

青少年は興奮する。
新しい感性を伝えてくれる。パソコン〈FM-7〉。マニアのために、新登場。
富士通
富士通株式会社 半導体統轄営業部 〒105 東京都港区虎ノ門2-3-13 TEL. (03)502-0161
●札幌営業所(011)271-4311 ●東北営業所(0222)64-2131 ●長野営業所(0262)26-8222 ●静岡営業所(0542)54-9131
●名古屋営業所(052)201-8611 ●大阪営業所(06)344-1101 ●広島営業所(082)221-2288 ●九州営業所(092)411-6311
マイコンスカイラブ●東京・虎ノ門(03)591-1091/591-2561 ●東京・秋葉原(03)251-1448/251-1449 ●大阪(06)344-7628/341-0486
●名古屋(052)221-6016●札幌(011)241-4185 ●広島(082)247-3949 ●仙台(0222)66-8711

3機種そろってエキサイティング。

左脳も右脳も興奮させる。

高性能のパソコンが手軽な価格で使いこなせる。―デビュー以来数ヵ月、FM-7に話題が集中しています。FM-8の先端技術を活かした豊富な機能と多彩なソフトウェア。数かずの魅力を携えて使う人を興奮させずにはおきません。

読みやすく使いやすい日本語表示機能。オプションの漢字ROMカードを本体に装着すれば、漢字、英字、かな、カナ、記号など3,418種を表示、印字。日本語ワープロとしても使えます。

入門用から実務まで、1台でOKです。ズとして豊富に揃えた周辺装置により、目的に合わせてシステム構成できます。個人のデータ管理や、ビジネス用にも充分対応。

簡易言語を標準装備。

●操作は簡単、BASIC言語の知識も不要です。操作は質問応答方式。簡単な命令を指示するだけです。プログラミングをまったく必要としない対話形式になっていますので、BASIC言語を知らない人でも手軽に使いこなせるよう設計されています。

●身近な家計簿や住所録から、オフィスの各種資料まで幅広く利用でき、作表、計算、検索、分類、ファイリングなどの作業に威力を発揮します。これらの機能を駆使して最終的に表形式のファイルにまとめることにより、身近なものからオフィス内の資料まで十分に対応できます。

●大量のデータの入力が可能。画面やプリンタへの出力も自在にレイアウトできます。 ひとつの表について、20項目まで設定でき、最大250文字のデータ入力が可能です。しかも、FM-7のメモリ容量の限界まで入力できます。(1表250文字で55枚まで可能)。一度入力されたデータは、その内容や分類の仕方を問わず、簡単な検索条件と、画面やプリンタへの出力フォーマットを指定するだけで、必要な資料を短時間に作成できます。

ご家庭のカラーテレビに接続して楽しめます。お手持ちのカラーテレビで美しいカラーグラフィック表示が楽しめます。市販のゲームソフト、プログラムの作成もOK。倍速モードも楽しめます。

F-BASICを標準装備し、盛り沢山の機能をサポート。FMシリーズ用に機能強化したF-BASICを本体内に実装。プログラミングが容易で、FM-8用の流通ソフトのほとんどがそのまま使えます。

サウンドクルージングを楽しめるサウンド機能も内蔵。シンセサイザ用LSIを内蔵。三重和音までの音楽演奏が楽しめます。ゲーム効果音もバッチリ。外部スピーカの接続も可能です。

ドット毎に8色まで色指定できるグラフィック機能。カラーグラフィック画面は640×200ドットの高分解能表示が可能。1ドット毎に8色までの色指定ができ、パレット機能で色交換も簡単です。

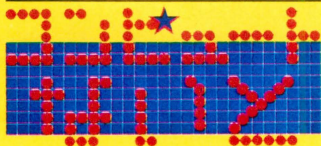
FM-7 MAZE

先端技術が夢中にさせる興奮パソコン。**新発売**

FM-

セブン

¥126,000(本体価格 簡易言語ソフト付)

パソコンとワープロが使いこなせるテレビ新番組

富士通提供「コンピュートないと」好評放映中

●放送日:毎月曜日、夜11時15分~11時45分 ●放送局:テレビ東京/テレビ大阪

●出演者:司会/相原友子:講師/石田晴久(東京大学教授)他に有名ゲスト、レポーターが登場

※テキスト「パソコングラフィックス入門」工学社より発売中(定価650円)

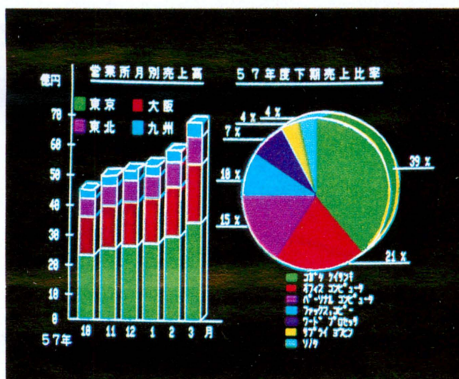
富士通のパソコンFMシリーズ。

ディスク搭載、16ビットが興奮させた。

オフコンにせまる本格派。EDP部門から現場まで本格的ビジネスユースに対応できます。
FMシリーズの高級機種として、FM-11が新登場。EXタイプはCP/M-86を標準装備。これによりCIS-COBOL(ANSI-74準拠)、Pascalなどの高水準言語を利用できます。そして、BASICは、使いなれたFM-8と上位互換性のあるF-BASIC。オフコンでのデータ処理を分散化したいというEDP部門と、今まで使っていたFM-8のBASICでデータ処理したいという現場双方で、同一機器の使用が可能になりました。

フロッピィディスクを標準装備、そして分離型。操作性、扱いやすさをさらに向上。FM-11は、本体、キーボード分離のセパレート・タイプ。本体とキーボードとの接続コネクタは、本体裏側から。しかもケーブル(カールコード)は2.5mの長さです。キーボードは操作性を重視したロープロファイル、シリンドリカル・ステップ・スカルプチャー設計。FM-11EX、ADの本体にはフロッピィディスクが1ドライブ標準装備され、さらに1ドライブ増設可能です。

日本語処理機能をいっそう充実。日本語処理をサポートしたCP/M-86の採用により、COBOLなどの高水準言語でも日本語処理が行なえます。もちろんF-BASICでも処理可能。カーソル位置に漢字表示でき、漢字の拡大、外字登録も可能、表示文字数は最大40字×25行、などなど、使いやすさは抜群です。(漢字ROMカード、オプション)

メモリは128KBを標準実装、最大1MBまで増設可能。メインメモリとして、128KB標準実装されており、さらに本体内の拡張スロットに増設RAMを実装すれば最大1MBまで拡張できます。これによりCP/M-86、MS-DOS、OS-9などのOSのもとで、大容量メモリを必要とする高水準言語を余裕をもって利用できます。

多彩な機能を発揮するCRT表示。最大640×400ドットのグラフィック表示は、キャラクタとの混在表示も可能。カラー表示は16色。最大12画面を瞬時に切り換えて表示できるマルチページ機能、色交換が簡単にできるパレット機能、上下さらに左右へも可能になったスクロール機能、などCRT表示機能がより多彩になりました。

簡易言語「FMCALC」を標準装備。
●コマンドは18種類。うち10種類がファンクションキーに設定されています。「FM CALC」は、FM-11の電源を入れると同時に実行され、画面に表が現われます。わずか18種類のコマンドで操作でき、そのうち10種類がファンクションキーに設定されているので、ワンタッチ操作ができ、さらに使い易くなりました。
●目的に合わせた処理ができ、会計業務、帳簿作成、統計処理など本格的ビジネス分野に利用できます。「FM CALC」は、プログラミングを知らない人でも「FM-11」を使いこなせるビジネスソフト。合計値や計算式など縦横のすべての関連数値を、データがひとつひとつ変わるごとに瞬時に計算します。表はスクロールによって、横52項目×縦128行まで作成できます。

ニーズに合わせてお求めやすい3タイプ。
●EXは、薄型フロッピィ1ドライブを標準装備。CP/M-86、F-BASIC、FM CALCを添付。
●ADは、薄型フロッピィ1ドライブを標準装備。F-BASIC、FM CALCを添付。
●STは、F-BASIC(ROMバージョン)。

*FLEX、UCSD Pascal、OS-9、CP/M-80・CP/M-86、MS-DOSはそれぞれTSC社、カリフォルニア大学理事会、マイクロウェア社、ディジタル・リサーチ社、マイクロソフト社の登録商標です。

ビジネス用途を大きく拡げる高級パソコン。**新発売**

FM-11
イレブン

■FM-11 EX ¥398,000(本体価格 簡易言語ソフト付)
■FM-11 AD ¥338,000(本体価格 簡易言語ソフト付)
■FM-11 ST ¥268,000(本体価格)

高級ホビーからビジネスまでの多才パソコン。

FM-8
エイト

¥218,000(本体価格)

富士通株式会社 半導体統轄営業部 〒105 東京都港区虎ノ門2-3-13 TEL.(03)502-0161
●札幌営業所(011)271-4311 ●東北営業所(0222)64-2131 ●長野営業所(0262)26-8222 ●静岡営業所(0542)54-9131
●名古屋営業所(052)201-8611 ●大阪営業所(06)344-1101 ●広島営業所(082)221-2288 ●九州営業所(092)411-6311
マイコンスカイラブ●東京・虎ノ門(03)591-1091/591-2561 ●東京・秋葉原(03)251-1448/251-1449 ●大阪(06)344-7628/341-0486
●名古屋(052)221-6016 ●札幌(011)241-4185 ●広島(082)247-3949 ●仙台(0222)66-8711

富士通パソコン用
キーボードカバー
KEYBOARD COVER
●美しくキーボードをカバーします
●デモ中に他の人の手を触れるのをガードします
FUJITSU MICRO 7
イニシャルプレゼント
キーボードカバーをお買いあげの方にもれなく、あなたのイニシャルシールをプレゼントいたします。
FM-7用 ¥3,800
FM-8用 ¥4,200
(送料 7・8用共¥500)
注文方法
通信販売だけです
発売方法
型番・郵便番号・住所・氏名・電話番号・イニシャルをはっきりお書きのうえ、下記へ現金書留または銀行振込みでお申しこみください。
総発売元
エイト電気株式会社
〒110 東京都台東区上野5-3-4 ☎03-831-5632(代)
振込銀行：北陸銀行 上野支店（普）NO.4032110
関西地区発売元
三栄電子株式会社
〒556 大阪市浪速区日本橋東2-10-2 ☎06-643-3833(代)
振込銀行：三和銀行 恵美須支店（普）NO.189879

EPSON

プリンタは、エプソン

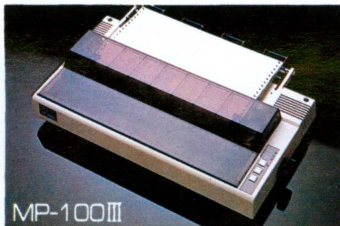
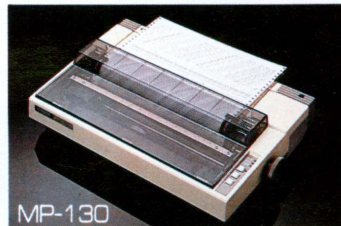
エプソンの最新鋭。超高性能・経済派プリンタ2機種、新登場。

FP-80 高速印字と多彩な文字バリエーションの強力インテリジェントターミナル

●ファンフォールド紙、ロール紙、レター用紙が使える可変ピンフィード、フリクション&トラクタフィード(オプション)方式●標準インタフェイス:セントロニクス準拠8ビットパラレル●インパクト・ドット・マトリックス●超高精度ビットイメージプリンティング9種類●印字速度、160CPS(テキスト時)●アンダーライン機能、スーパー/サブスクリプト文字、強調文字など多彩な文字種に加え、国際文字印字(9ヵ国対応)、ダウンロード文字(ユーザー定義可能文字)、さらにプロポーショナル印字も可能。また、すべての文字種は同一行混在可。左右のマージン設定、水平タブ、垂直タブなどの機能によりフォーマティングが自由自在。¥149,000/¥152,800(PC-8001専用)/¥153,800(PC-8801/9801専用)

RP-80 MPの技術が生んだ高性能の経済派プリンタ

●4″~10″のファンフォールド紙が使えるトラクタフィード方式●標準インタフェイス:セントロニクス準拠8ビットパラレル●インパクト・ドット・マトリックス●単密度480ドット/行、倍密度、倍速倍密640ドット/行、CRTグラフィックス640、720ドット/行、4倍密度1920ドット/行の高精度ビットイメージプリンティング6種類●印字速度:100CPS(テキスト時)●アンダーライン機能、スーパー/サブスクリプト文字、強調文字など多彩な文字種と豊富なファンクション
¥89,000

14×18ドット、80桁普及型漢字プリンタ
●1台で3役、漢字(双方向印字)/通常文字(双方向最短距離印字)/ビットイメージ(単方向印字左→右)●4″~10″のファンフォールド紙が使える可変スプロケットフィード方式●JIS第1水準準拠、漢字・記号約4,000字装備 ¥189,000/漢字コード対応表 ¥500

136桁ビットイメージプリンタの普及版
●ファンフォールド紙、ワンシートが使えるフリクション&トラクタフィード方式●桁数は68桁(拡大)、136桁(標準)、116桁(縮小の拡大)、233桁(縮小)の4種類●単密度816ドット/行 倍密度1632ドット/行の高精度ビットイメージプリンティング●アンダーライン機能、スーパー/サブスクリプト文字、強調文字など多彩な文字種と豊富なファンクション ¥189,800/¥192,800(PC-8001専用)

高速136桁ビットイメージプリンタの本格派
●135字/秒の高速プリンティング●底部からの用紙(ファンフォールド紙)挿入もできる2ウェイ(背面・底面)紙送り方式採用の本格ビジネス仕様●1行816ドットのビットイメージプリンティング ¥228,000

本格的ビジネスユースに最適の高性能24×24ドット、136桁漢字プリンタ ●ファンフォールド紙、ワンシートでB4横サイズまで使える可変スプロケット&フリクションフィード方式●漢字90桁/行、アルファベット・カナ・記号136桁/行●JIS第1水準準拠約4,000字、JIS第2水準準拠約7,000字の漢字ROMを内蔵、JISコードでハンドリングが容易●単密度1224ドット/行、倍密度2448ドット/行の高密度ビットイメージプリンティング ¥510,000(JIS第1水準漢字ROM内蔵)
¥550,000(JIS第2水準漢字ROM内蔵)

DP-20デイジーホイールプリンタ

鮮明な印字と簡単な操作のインパクトプリンタ
●デイジーホイール採用により、ワードプロセシング等に最適●ホイールは600万文字と高寿命●カセット式デイジーホイールおよびリボンメカニズムの採用により、活字・リボン交換がワンタッチ●文字/行:132文字(1/10インチ)、158文字(1/12インチ)、198文字(1/15インチ)●用紙幅:16.5インチ(420㎜)●キャリッジ送り:10文字/インチ、12文字/インチ、15文字/インチ●ペーパーフィード方式:両方向フリクションフィード
¥230,000(シリアル仕様)/¥210,000(パラレル仕様)

各種パソコン対応表

		RP-80		FP-80		FP-80 (PC-8001専用)		FP-80 (PC-8001MKII、PC-8801 PC-9801専用)		MP-100Ⅲ		MP-100Ⅲ (PC-8001専用)		MP-100Ⅲ (PC-8801、PC-9801専用)		MP-100Ⅲ (M20/23専用)		備考
		COCD	HC	CODE	HC	CODE	HC	CODE	HC	CODE	HC	CODE	HC	CODE	HC	CODE	HC	
EPSON	QC-10	◎	●	◎	●	○	●	○	●	◎	●	○	●	○	●	—		
	QC-20	◎	●	◎	●	○	●	○	●	◎	●	○	●	○	●	—		
	HC-20	◎		○		○		○		○		○		○		—		#8145により接続
NEC	PC-6001	○	●	—		—		—		—		—		—		—		
	PC-8001	◎	◉	○	◉	◎	◉	◎	◉	○	◉	◎	◉	—		—		#8135によりハードコピー可
	PC-8001MKII	◎	●	○		◎	●	◎	●	○		—		◎	●	—		
	PC-8801	◎	●	○		◎	●	◎	●	○		—		◎	●	—		
	PC-9801	◎	●	○		◎	●	◎	●	○		—		◎	●	—		
三菱	MALT116	○	●	○	●	—		—		★○	●	—		—		—		
ソード	M20/23	○		○		—		—		○		—		—		◎		
タンディ	TRS-80	○		○		—		—		○		—		—		—		
アップル	APPLEII	◎	◉	◎	◉	—		—		◎	◉	—		—		—		インテリジェントパラレルインターフェイスによりハードコピー可
東芝	パソピア	◎	●	○		◎	●	◎	●	—		—		—		—		
備考				ダウンロード文字により各コンピュータと文字コード表を一致させることができます。														

CODE:文字コード表 HC:ハードコピー ◎文字コード表一致 ◉オプションによりハードコピー可 ○グラフィックキャラクタ不一致 ●ハードコピー可

パソコンのディスクユース時代に、先駆のターミナルフロッピーTF-20。

新時代のスリムなフロッピーディスクドライブ装置TF-20。両面倍密度322K×2デッキを内蔵して142,000円(標準タイプ)

〈TF-20の特長〉薄型・軽量・コンパクト:幅が従来品の1/3の新開発薄型ミニフロッピーディスクドライブ2基を搭載、2ドライブのミニフロッピーディスクユニットとしては驚異的な小型化を実現しました。パソコンが小さくなってもフロッピーが…という悩みを解決したスペースセーブぶりです。●**高い操作性**:ディスクを挿入しプッシュボタンを押してセット完了。ディスクを取り出すには、押し込まれているプッシュボタンを更に押すと約1cmディスクが取り出し口よりポップアップ。簡単でしかも確実な操作を可能にしています。●**ハイコストパフォーマンス**:便利なのはわかっているけれど価格が高すぎて…。TF-20は自社開発ディスクドライブを採用し、高い信頼性、性能を確保しながらトータルコストダウンに成功しました。

〈仕様〉インタフェイス:シュガートSA400コンパチブル●インタフェイスコネクタ:FRC2-C34(DDK)●記録方式:FM、MFM●トラック間セクタ時間:15ms●トラック密度:48TDI●トラック数:80トラック/両面1ドライブ●最大記憶容量:1Mバイト(アンフォーマット)●電源:AC100V(50/60Hz)●消費電力:40W以下●外形寸法:120W×350D×165H(㎜)●重量:7.0kg●価格:エプソンHC-20専用機¥177,000、NEC PC-8001・8801専用機¥166,000、富士通MICRO-8専用機¥163,000

エプソン株式会社 本社/〒399-07長野県塩尻市広丘原新田80 ☎02635-4-0271 ●詳しい資料をご希望の方は、はがきに住所、氏名、年令、職業、製品名をお書きの上、もよりの支店・営業所までお申込みください。●東京支店/〒160 東京都新宿区西新宿2-4-1 新宿NSビル内私書箱6026号 ☎03-348-6801・大阪支店/〒530 大阪市北区西天満4-14-3 住友生命御堂筋ビル11階 ☎06(365)5071・名古屋営業所/〒460 名古屋市中区金山1-12-14 ☎052-331-5486・札幌営業所/〒060 札幌市中央区北1条西2丁目札幌時計台ビル6階 ☎011-222-2821・仙台営業所/〒980 仙台市本町1-12-12 ☎0222-63-3691・長野営業所/〒390 松本市深志2-2-9 加納ビル2階 ☎0263-36-7251・広島営業所/〒730 広島市中区幟町3-1 第3山県ビル4階 ☎082-247-1685・福岡営業所/〒812 福岡市博多区博多駅前3-4-8 ☎092-471-0761

キミの頭脳は高感度すぎて
時代はキミをクレイジーと呼ぶが、
私たちは、そのスーパークレイジーぶりに
期待している。
クレイジーと天才は紙一重、
1時代の差があるだけだ。
主役はいまスーパークレイジーの時代。

スーパークレイジーたちの祭典

PASOL SOFT CONTEST

ヴォルテージ

ぱそる ソフトコンテスト
賞金総額 400万円

通信販売、店頭販売 8大特典

沖縄の旅10名様ご招待

期間中1万円以上お買上げのお客様を抽選で沖縄の旅にご招待いたします。

(期間/58年5月20~9月30日)

お買上げ金額ごとに豪華プレゼント

1. 30万円以上 ポケコンPC-1245
2. 20万円以上 ゲーム機器
3. 10万円以上 テープケース
4. 10万円未満5万円以上 生テープ5本

(期間/58年5月20日~9月30日)

現金・分割とも**超特価**奉仕

1万円以上お買上げのお客さまを、ソフトコンテスト受賞イベントへ抽選でご招待いたします。

(期間/58年5月20日~9月30日)

機能拡大、愉しみもまた拡大

●PC-8001mkII(本体) ~~123,000円~~
●FM-7(本体) ~~128,000円~~
●MZ-731(本体+カセット・レコーダー+カラー・プロッタ) ~~128,000円~~

(PC専用)

品名	価格
フロッピーディスク(PC-80S31, NEC)	~~168,000円~~
シンセサイザー(CMU-800, アムデック)	~~65,000円~~
シンセサイザー(PCS-8007, パックエレクトロ)	~~24,800円~~
音声発生(Mr.ボイス, テックメイト)	~~49,000円~~
音声発生(読み上げさん, 和幸舎)	~~35,000円~~

(PC, FM兼用)

品名	価格
プリンター(GP-250F, 精工舎)	~~59,800円~~
プリンター(GP-700M〔カラー〕, 精工舎)	新製品
プリンター(GP-550E〔漢字〕, 精工舎)	新製品
プリンター(RP-80, EPSON)	~~89,000円~~
プリンター(FP-80, EPSON)	~~149,800円~~

品名	価格
プロッター(マイプッロットJr, 渡辺測器)	~~99,800円~~
フロッピーディスク(TF-20, EPSON)	~~142,000円~~
カラーCRT(FTC-1416, 東映通商)	~~63,000円~~
カラーCRT(FTC-1425〔高解〕, 東映通商)	~~105,000円~~
デジタイザー(K-510, 関東電子)	~~148,000円~~

募集要項

応募作品

①ゲームソフトおよび教育用ソフト
（教育用ソフトはゲーム要素の強いもの）
②ゲームおよび教育用ソフトのアイディア
両部門とも、自作の未発表作品に限ります。

応募条件

使用機種は、メーカーは問いませんが、特殊なものは除きます。使用言語はBASICまたはマシン語とし、プログラムはカセットテープ、ミニディスケットのどちらでもかまいません。

応募方法

①作品には、使用機種、簡単なプログラムの紹介(名称等)と操作方法ならびにプログラムリスト、サンプルデータ等を添付のこと。
②アイディア作品については、3画面以上のカラーイラストとシナリオを添付すること。
③応募者は作品に、氏名・住所・電話番号・年令・職業（学生の場合は学校名）を明記。また簡単な自己紹介を添付のこと。

応募締切

昭和58年9月10日（当日消印有効）
発表
入選者は10月末に直接本人に通知するほか、店頭、本誌12月号および各専門誌上にて発表します。

賞

総額400万円

●ソフト部門

最優秀賞1点	賞金	150万円
優 秀 賞2点	賞金	各50万円
佳　作5点	賞金	各15万円

●アイディア部門

最優秀賞　1点	賞金	30万円
優 秀 賞　3点	賞金	各7万円
佳　作12点	賞金	各2万円

審査

当社開発部、各メーカー担当者およびゲスト審査員により、厳正な審査を行ない、各賞を決定します。
●ゲスト審査員　所ジョージ・石川ひとみ

作品の送り先と問合せ先
〒107 東京都港区赤坂1-5-11　新虎ノ門ビル
㈱ぱそる開発部「ソフトコンテスト」係
☎03(588)-1717　担当　辻本
※入賞作品の版権はすべて当社に帰属します。また応募作品は原則としてお返しいたしませんのでご了承ください。返却希望者は、郵送料を添えてご応募ください。

は12月にやってくる。

受賞式は華々しくにぎやかに開催します。お祝いに駆けつけるゲストの顔ぶれもすごい。楽しみは12月までおあづけだけど、待てるかな。

●ゲスト 石川ひとみ

●ゲスト審査員 所ジョージ

●司会　木藤隆雄
（ドレミファドン レギュラー）

日時　昭和58年12月10日（土）
P.M1:00～4:00
会場　「ぱそる」店内

受賞式の内容が一部変更になる場合もあります。あらかじめご了承ください。

特典5 配送費無料

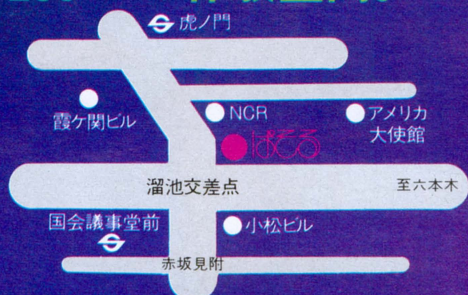
特典6 修理費（配送費とも）無料

特典7 日本信販との提携ローンで最長36回まで分割可能

特典8 テレフォンサービス実施
技術指導、ご質問等お気軽にお電話ください。
TEL 03(588)1717
（平日a.m10：00～p.m7：00
土曜a.m10：00～p.m5：00
＊日・祭日はお休みです）

全国の皆さまへぱそる通信販売

電話またはハガキにてお申し込みください。
どこへでも即納（配送無料）します。ハガキでお申し込みの場合 ①機種名 ②お支払い方法（現金またはクレジット回数）③氏名・住所・電話番号・年令・職業をご明記ください。

ゆったり、たっぷり
260m² の体験空間。

本社・ショップ 〒107東京都港区赤坂1-5-11
（新虎ノ門ビル1Ｆ） ＴＥＬ03(588)1717(代)
和歌山営業所 〒624 和歌山県海南市藤白758番地
富士興産(株)内 ＴＥＬ07348(2)7332(代)

（FM専用）

フロッピーディスク(薄型フロッピー, 富士通)	167,000円
プリンター(MB27404, 富士通)	89,000円

（MZ専用）

カラーCRT(MZ-1005, シャープ)	69,800円
ジョイスティック(Joy-700, 九十九電気)	5,800円

カタログご希望の方は、ハガキに氏名・住所・年令・職業をご明記のうえ、カタログ請求券を貼ってお申し込みください。

カタログ請求券
oh! FM6月号

FMシリーズ

ソフト フェア

(25種類)

全国200店のSHOPで展開!

5/20〜6/23

◉詳しくは、お近くのSHOPへお問合せください◉

株式会社 日本ソフトバンク

東京都千代田区九段南2-3-11　靖国九段南ビル2F　☎03(263)3690(代表)

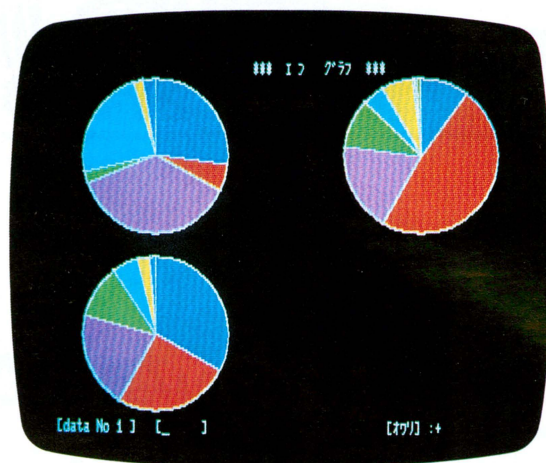
富士通パソコン情報誌

Oh!FM 第3号

目次

グラフィックソフトパッケージ

パソコンもほんとうに楽しくなってきました。近頃のマシンといったら，みな美しいグラフィック画面を持っていて，リアルな絵で私たちに語りかけてくれます。

そこでOh!FMでは，美しいグラフィック機能をふんだんに使ったソフトをご紹介することにしました。

◇グラフパッケージ◇

日本マイコン　FM-8
●50,000円　Disk

データの視覚化はパソコンが最も得意とする仕事の一つだ。グラフ化された数値は具体的な説得力をもち，正確な認識を保証する。このグラフパッケージは，様々なグラフをサポートしている。帯グラフ，円グラフ，棒グラフをはじめ，レーダーチャート，カテゴリー別グラフ，棒グラフ，折れ線グラフである。ディスクファイルも使え，大量のデータが容易に短時間で取り扱える。またハードコピー機能は，資料としてまとめる際にとても便利だ。

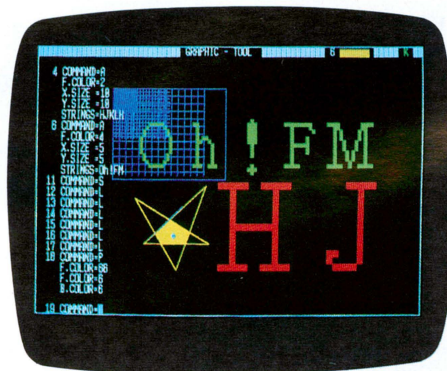
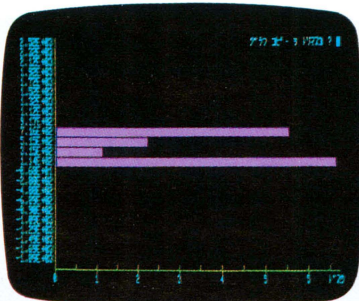
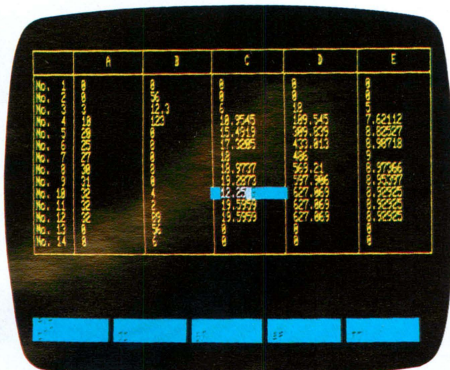
データの正確な把握を求める人にはおすすめできるソフトだ。

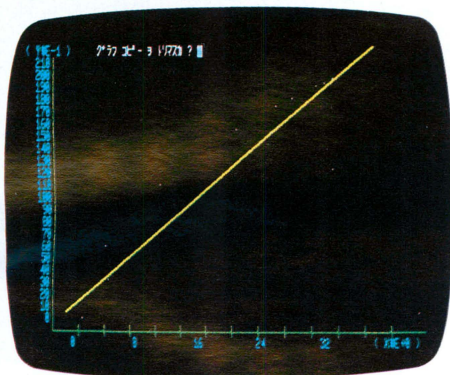
◇GT◇

ニチコン　FM-11
●10,000円　Disk

最近注目されている誰にでも使えるソフトに簡易言語と呼ばれるものがある。BASICの持つ命令を，BASICを知らない人でもすぐに使えるように，目的を限定し，簡略化したものだ。

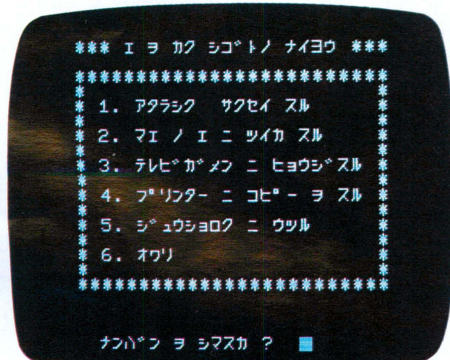
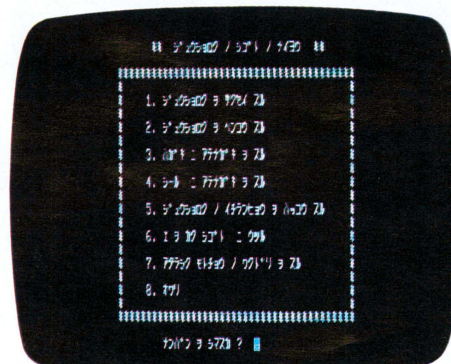
このGTは，グラフィックを活用することを目的とした簡易言語だ。BASICのプログラムで絵を描くのでなく，画面上に現れるカーソルを動かしながら，線を引いたり，円を描いたり，色を塗ったりできる。俗にグラフィックエディタと呼ばれるソフトの一つだ。GTのコマンドは強力で，線を引くコマンドはもちろんのこと，パレットの変更，漢字の拡大表示，メッシュのコマンドまでもサポートしている。

グラフィックを使うことなら何にでも応用できるが，ポスターやディスプレイの設計や配色シミュレーションには特に有効であろう。

◇TECHNICA◇

マイコンセンターウエノ　FM-8
●9,500円　テープ

パソコンのグラフィック機能はグラフ作成にも威力を発揮する。このテクニカは，グラフ作成というよりも実験データの処理のためのツールだが，相関プロットなど視覚的な機能も多くサポートしているため，ここで取り上げてみた。

テクニカは科学分野で生じる複雑多岐なデータを利用しやすい形式に自由に処理し，レポートの作成を容易にすることができる。特徴を並べると，

①スクリーンエディット機能により手軽にデータシートを作成できる。
②データシートの分類，集計および計算式によるデータの変換ができる。
③ユーザー定義の計算式は，四則演算はもちろん，SIN，COS，TAN，EXP，LOG，SQR，X^Y などの関数も使用でき，数式どおりに設定できる。
④利用度の高い統計計算はコマンドとして用意されている。標準偏差，加算平均，自乗平均，総和，母分散，不偏分散，標本標準偏差，相関係数，最小自乗近似などである。
⑤XYプロット，度数分布グラフ，最小自乗グラフ，2項相関プロットのグラフ機能を持っている。

以上の特徴をみてわかるとおり，科学的なデータ処理には十分な機能を備えている。スクリーンエディットによるデータの入力も操作性よく，すぐにでも活用できるソフトである。めんどうな科学計算やグラフ化にてこずっている方にはぜひとも一度試してもらいたい。

アラレちゃん——Ⓒ鳥山明，集英社・少年ジャンプ「Dr.スランプ」より

◇ピクチャーメール◇

近畿コンピュータサービス　FM-7,8,11
●5,000円　テープ

パソコングラフィックスを楽しく使うと，画面にまんがを描いたり，それを葉書にハードコピーして友だちに一風変わったテクノっぽい手紙を出したりすることができる。このピクチャーメールは，そんな遊びがだれにでもできるようにする，ありがたいソフトだ。

“エヲカクシゴト”では，プログラムなんか関係ないし好きな絵を書くことができて，それに葉書に画面上の絵をプリントすることができる。

“ジュウショロクノシゴト”では友達の住所を管理したり，葉書にあて名書きしたりできる。

この二つのシゴトをするとコンピュータでつくった手紙ができあがる。暑中見舞ではちょっと変わったパソコン葉書で目立ってみませんか。

ていることの方が確からしい。

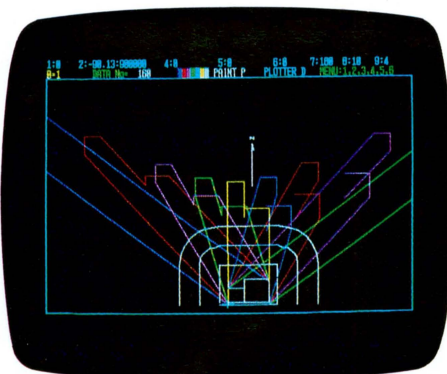
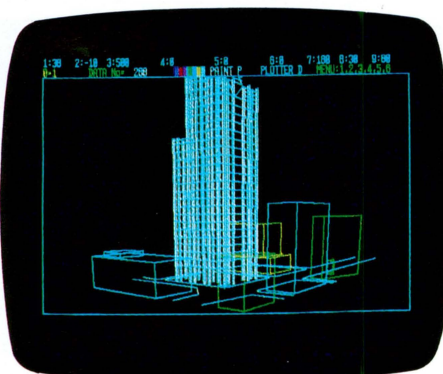
何かゲームでも作ろうとして，ちょっとした小さなパターン，たとえば古くなるがインベーダゲームのインベーダのパターンが欲しくなってもそう簡単には作れない。BASICでPUTとGETという命令があるので，この命令を生かす形でパターンをつくりたいがどうにもうまくいかない。そういう人のためには，このソフトを紹介するのが一番だと思う。

使い方は簡単だ。画面右に操作法が一覧され，左側には，ドット単位のます目が拡大されて表示されている。操作手順に従ってカーソルを動かしてドットの色を決めていけば，あとは自動的にPUT文用のデータをつくってくれる。データができたら，これをBASICの文中で配列に読み込ませ，その配列でPUTすれば，そのパターンが画面上に現れるはずだ。

説明，操作性ともによくできているから，はじめての人でも使いやすいソフトだ。

◇パターンエディタ◇

テクノソフト　FM-7,8
●3,300円　テープ

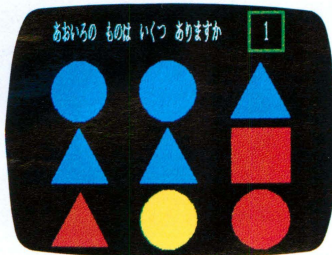
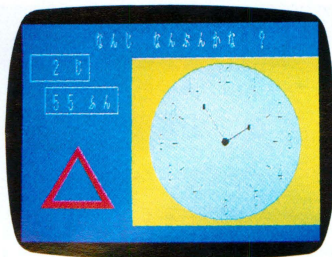
FMシリーズは絵を描くには十分緻密なグラフィック画面を持ってはいるものの，ユーザーがみなそれを十分に使いこなしているかというと，答えは否定的だ。その原因を考えてみると，個人のイメージの不足というよりも，使いやすいソフトが不足し

◇PERS-F1,F2◇

マール社プランニング・センター
FM-8,7　●80,000円　Disk

コンピュータを使ったグラフィックの分野に，CADというものがある。Computer Aided Designの略で，コンピュータをインタラクティブに用い，設計を容易にするシステムのことだ。今までは大型コンピュータの領域だったが，パソコンの性能の向上に伴い，個人レベルでの設計はある程度パソコンにまかせることができるようになった。

ここに紹介するPERS-F1は，透視図を描くソフトである。陰線の処理はできないが，対象物の三次元データを入力すると，それを任意の方向から見た透視図を作成してくれる。プロッタを用意すれば，CRTでは表現できない繊細な線も描け，製図として充分通用する。

これから設計分野へのコンピュータの導入はますます盛んになるであろう。製図，デザイン，レタリングに専門的にたずさわろうという方には入門用として試みることをおすすめする。時代を先取りした新しい体験ができると思う。

なお，このソフトに関しては，マール社からパソコンで透視図を描くという詳しい解説書が出ている。具体的な使用例，応用例が多数掲載されており，透視図の基本から解説してある。詳細はまずその本を見ていただきたい。

プログラム名	左右の理解
数のしくみ	数の合成 1
色と図形	数の分解 1
あわせていくつ	いまなんじ 1
くらべっこ	いまなんじ 2
数の合成と分解 1	数の合成と分解 2

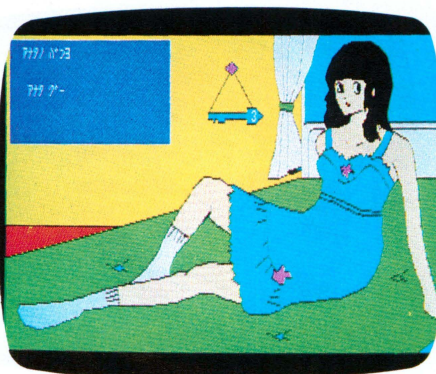
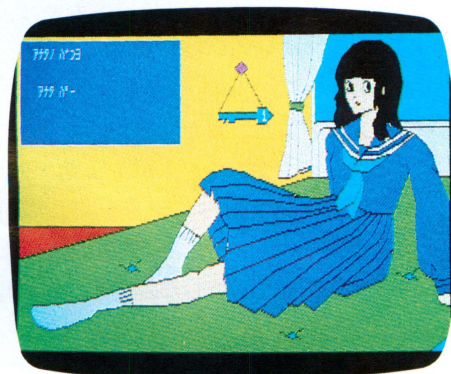
◇スキップⅡシリーズ◇

鈴木楽器販売　FM-7
●各3,000～4,000円　テープ

コンピュータの応用の一つにCAI(Computer Aided Instruction)という分野がある。これはコンピュータを教育に活用することで，コンピュータと生徒が対話形式で生徒のレベルに合わせて学習できる。

鈴木楽器の教育用ソフトはよくいわれるCAIとは少々異なり，幼稚園で数のかぞえ方や大小の概念を教えるときに用いる教材のかわりにパソコンを使おう，という発想から生れた。

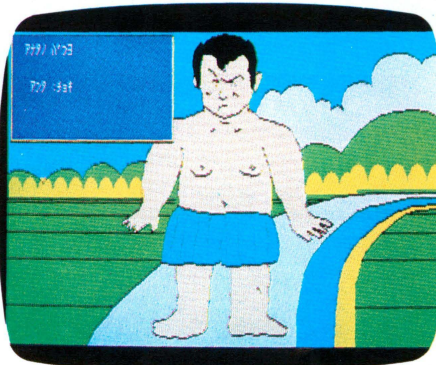
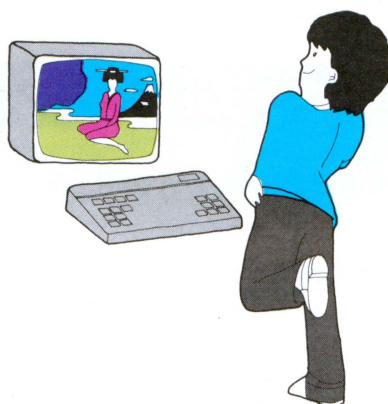
写真のとおり，時計やかたちの教材が，そっくりCRT上に表示されている。特にパソコンを使った利点をあげると，多彩なアニメーションとサウンドにより楽しく学習を進めることができる，ということと，テレビとは違い生徒の要求に応じて画面を操作することができるということだ。このソフトはこの点を十分重視して作られており，実績を積んでいるだけに完成度は高い。

現在，テレビの画面を自由にレイアウトでき容易にアニメーションが作れる機器はパソコンしかない。このことを考えると，鈴木楽器の試みはまだまだ発展性を秘めている。幼児教育にたずさわっている人には一度は参考にしてもらいたいソフトだ。

◇セーラー服と野球拳◇

CSKソフトウェアプロダクツ
FM-8,7　●3,800円　テープ

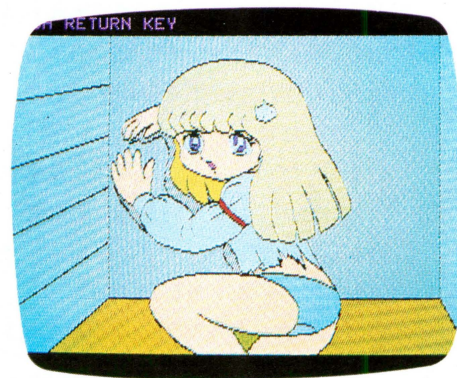
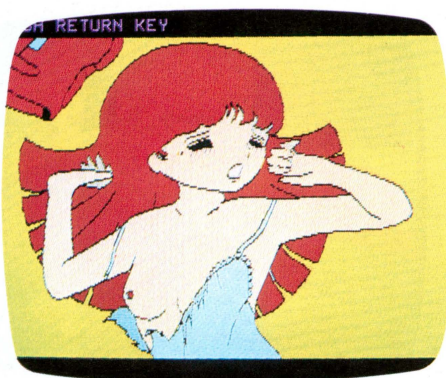
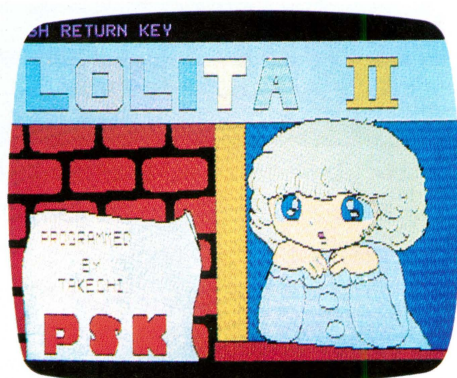
はい，おまたせ。ちかごろはやりの野球拳。野球拳といったら，昔は吉原芸者相手にお金持ちのおじさんが酔いにまかせてやる遊びだったのだが，時代は変わり今やコンピュータ時代。マイコンのファイングラフィックスを有効に使ったソフトとしてにわかに注目されはじめた。

で，何がどうなるのかというと，めっぽう簡単。グー，チョキ，パーのうち1つを選んでキーを押すだけ。あなたが勝つと，美女がだだもこねずにすぐ脱いでくれる。もちろん負けたら，あなたも素直に脱ぐ。少々変態ぎみだが，でもこれがまた40代のおじさんに受けてもうたいへん。お相手をしてくれる女の子もバラエティに富み，女性用の男性ヌードまで飛び出す始末。

コンピュータの普及に大いに貢献しているこのソフト，まだまだブームは続く。

◇芸者ゲーム◇

CSKソフトウェアプロダクツ
FM-8,7 ●3,800円 テープ

またまた出た野球拳。今度は，富士を見ながら芸者さんを脱がそうというもの。なかなか凝った設定，パソコンあってこそできるぜいたくだ。

さて，野球拳で困ることは，操作する側が規則を守らないこと。そう，自分が負けたのにコンピュータが知らないことをいいことに，何も脱がずに続けること。でも考えてみると，自分も脱いで相手も脱ぐからこそ野球拳の味が出るというもの。一度は一対一の真剣勝負で本場の味を確認していただきたい。その緊迫感といったら，たまらない。

◇ロリータII◇

パソコンショップ高知 FM-7
●7,000円 Disk

あのアドベンチャーゲームのなぞ解きの魅力と，野球拳に代表されるCRT上の愛らしい美少女たちが一つになって，ロリータIIが完成した。

ストーリーを紹介しよう。あなたは，黒いサングラスに白いマスクで顔を隠し，よれよれのコートを着，誰が見てもちょっとくせのある姿だ。あなたは今，袋小路の角で下校中の少女たちをうかがっている。ポケットにロープとハンマをかくしつつ，あの子がいい，いやとなりの子はもっとすてきだと，よだれたらたら。もしめがねとマスクをはずしたら，それはまぎれもなく小羊をねらう狼の顔に違いないだろう。

あなたはもうこわいもの知らず，ときどき現れる警官の前では静かにしているものの，少女だけのときは片端からレイプしてしまう。抵抗されると面倒だから，まず縄でしばり，ハンマで気を失わせて………。

あとはご自分でどうぞ。ただあくまでも他人のいないところでやろう。ブラウン管を見つめるあなたの顔は，きっと狼以上だろうから。

マシン語特集

マシン語って何んなのですか？

―マシン語のリストを目前にあ然と立ちつくすあなたへ―

慶応大学メモリーマイコンクラブ　田奈　真実

ある日のことでした

とある雨の日のこと，じとじと降る雨で得意のバトミントンもできず，フラストレーションの累積が緊張した雰囲気を醸し出しているなか，あの忌まわしい電話のベルがなりました。遠くの受付が電話をとったのでしょう，ベルが止まります。この時間は通称「もしもし電話」質問受け付けの時間帯なんです。果たしてそのベルは営業・出版にまわらずして僕らのところに回ってきました。

「もしもし，お電話かわりました。」

「あの―，FMについてお聞きしたいんですけど。」

というその声を聞いてぶっ飛びました。この声で一気にフラストレーションが揮発し，雰囲気までも明るくなったのです。それはまぎれもない女性の，それも若い声だったのです。下心が前面に出ぬよう慎重に言葉を選び応えます。

「どういった内容でしょうか。」

「マシン語って何なのですか。」

この質問が男からのものであれば，ほとんど怒りに近いものが込み上げてくるのでしょうが，女性からの電話ですっかり混乱し，どう答えてよいものか考えてしまいました。質問の電話は雑誌の内容に関するものしか受けていないのです。それにこのような漠然とした内容では電話で説明できるはずもありません。そこで混乱しながらもこう答えました。

「ちょっと質問の内容が難しいもので，電話ではお答えしにくいのです。今度 Oh! FM でお答えを載せますから，それで御勘弁願えないでしょうか。」

質問された，若くてかわいく純情でやさしく賢く身長が170cm以下でちょっとぽちゃっとしている女性の方には，それで納得いただけたようです。

というわけで，今回「マシン語とは何か」という非常に基本的ではあるけれども重要な話題にふれてみようと思います。

マイコンの中にはCPUが……

マシン語と聞いて何を連想されるでしょう。速い，そしてゲーム，そして**図1**のようなリストではないでしょうか。ではなぜ，マシン語のプログラムはこのような形で表されるのでしょう。

マイコンの中心部にはCPUと呼ばれる部分があって，ここが計算とか表示とか命令の解釈とか何とか，全て知的と思われることをしているのです。で，このCPUという部分，デジタルで考え，デジタルで指令しています。デジタルというのは，「数値の」という意味で，よって数を用いて命令とか計算を行うということになりますね。計算が数を用いて行われる，というのは当たり前で，だから計算なんだという声も聞えてきそうです。そこでちょっと納得できないのが，命令を数で行うという点だと思いませんか？

図1　マシン語のリスト

Add	+0	+1	+2	+3	+4	+5	+6	+7	Sum
9000	E1	96	E1	9C	D9	A1	CD	BE	:F9
9008	D0	14	D1	F5	D1	ED	CA	A5	:D7
9010	D1	6C	D1	37	CE	8B	D6	BD	:31
9018	DD	90	E6	24	DD	2E	CB	03	:50
9020	E2	56	AB	74	9E	BD	9F	0D	:5E
9028	94	AE	CF	D7	CA	B1	BF	4D	:6F
9030	B0	F5	B1	23	93	F8	E9	53	:40
9038	E9	6E	E6	B6	DE	E3	DF	91	:24
9040	E0	1D	EC	33	EF	6D	EF	70	:D7
9048	F1	FE	E5	0E	8D	E2	9D	8E	:7C

コンピュータってやっぱしデジタル

命令が数で行われる例を考えてみましょう。1＋1の計算をしてみることを考えると，普通の言葉では，

　1と1をたして結果を教えてね。

ということになります。電卓だと，

　1＋1＝

と押せば，望みの結果が得られます。命令も数でやるとなればどうなるでしょう。仮に「たせ」という命令を「1」，「結果を表示せよ」を「2」という数で表すと，

　1 1 1 2

となります。これは，「イッセンヒャクジュウニ」ではなく，「イチイチイチニ」と読んでください。

何んとなく命令も数で表現できそうですね。では11＋1＝はどうでしょうか。

11112

これでは1＋11＝とも，1＋1＋2ともとれます。つまり，純然たる数と，命令を表現した数が混然としてしまいます。

ではどうしたらよいでしょうか。最初に考えつく解決法は，

11010102
11 ＋ 1 ＝

と全て2桁とすることです。しかし，これでは3桁，4桁と大きな桁の計算では，命令も3桁，4桁となり，またその都度規則を変えるのも効率が悪いし，スマートさにも欠けます。そこで，IDというものを使うことになります。

学生証もID，でもここでは……

IDとは，身分証明書などを指しますが，コンピュータ用語では「識別子」と訳されています。識別するためのしるしのことなのです。

電卓の場合を考えてみましょう。

[1][1][＋]

と押せば，11が入り，その次の数を待つことになります。3桁，4桁…でも同様です(と言っても上限はありますが)。この[＋]が「たせ」という命令と同時に，数はここまでというIDの役割を果たしています。

次にコンピュータの場合を考えてみましょう。「たせ」という命令も数で表現されますから，電卓と同じようにはいきません。そこで頭に，次に何桁の数がくるかというIDを付けます。仮に3は1桁，4は2桁，5は3桁の数（正確に言えば数値）と規定します。すると，11＋1の計算は，

と表せます。このような方法によって何桁の数でも表現可能となります。

何かおかしいですか？　確かに5で3桁，6で4桁……9で7桁と考えれば，7桁が限界のように見えます。でもちょっと頭を使えば，何桁でも表現できることに気づかれるでしょう。

ますますコンピュータらしく

以上述べた中では，コンピュータと電卓の違いがはっきり出ていません。むしろ，電卓を使った方がよっぽどましです。コンピュータのコンピュータたるゆえんは，仕事の手順を覚えていることです。判断し，それにより別の手順を行うことにあるのです。それでは，そのような仕事について考えてみましょう。

「人間が入力した数を2倍して表示し，それを繰り返す。」電卓では，その都度[×][2][＝]を押すか，置数計算のできる電卓でも[＝]を押し続けねばなりません。

コンピュータの場合を考えてみましょう。まず，新たに命令を考える必要があります。「人間に数を入れてもらう」命令を6，「かけ算」を7，「計算し表示するが終了しない」命令を8，「最初に戻れ」を9としてみます。すると，

となります。

次は2をどんどん2倍してみましょう。

3　2　7　3　2　8　?
　　　↑

うーん，困ってしまった。8の次はどうすればいいのでしょう。はじめに戻らずに矢印のところへ戻りたいのです。どうしたらいいのでしょう。

ついにアドレスの話

これを解決するには，アドレスという言葉を理解しなければなりません。

皆さんもいろいろな本で聞いたことがあるのではないでしょうか。アドレスとは住所のことで，データのしまってある場所の住所をいうのです。このデータという言葉もよく聞くものでしょう。

通常，データというのは資料といった意味で使われます。いわく，「データが足りなくて結論が出せない」，「データの取り方がおかしい」，「データが出一た」等々。コンピュータでは，広義の意味として，記憶している内容のことを言い，狭くは，命令などで処理されるものを言います。ここでは広い意味でデータを考えることにします。つまり，前にだした，「2倍にする」の手順（プログラムのことだね）はデータであるわけです。

673289

の，6とか7とかそれぞれの数字はデータなのです。で，コンピュータではこれらのデータの記憶場所をアドレスと呼んでいます。上の例だと，

アドレス	0	1	2	3	4	5
データ	6	7	3	2	8	9

と考えることができるでしょう。0番地が6，1番地が7，2番地が3というようにです。コンピュータは，このアドレスの順番にデータを読んでいって，実行をしていくわけです。

上のように，アドレスも数字で表します。ま，この辺がデジタルコンピュータのデジタルと呼ばれるゆえんで，全て数字で扱うというか，むしろ数字しか扱えないというのか……。

ジャンプッしちゃう

と，ここまできて，「どんどん2倍にする」が実現できそうになってきました。

さて，

327328?

でしたね，悩みはじめたのは。ここでは7の位置に戻ればよいのですから，

アドレス	0	1	2	3	4	5
データ	3	2	7	3	2	8

2番地へ行けばよいことになります。この「行く」というのをコンピュータではジャンプと言っていて，だから，この場合最後で2番地へジャンプすればいいんです。では0をジャンプ命令にしましょう。0の次の2つの数字をアドレスと考えてジャンプするとするのです。

327328002

ほら，できてしまった。これで望みの動作

をしてくれるんですよね，コンピュータが。

実は これが マシン語

以上説明したのがマシン語の**原理**なんです。すべてが数字で表されていて，それ以外何んにもない。パッと見たらどれが命令で，どれが数値なのか，プログラムがどんな意味で，どんな動作をするかわからない。ね，マシン語でしょう。確かにマシン語なんだよね。

例えば次の例で，

アドレス	0	1	2	3	4	5	6
データ	4	3	0	2	3	1	2

こんなプログラムがあったとします。0番地から実行を始めれば，30＋1＝で31という結果が出るんですが，1番地から始めると0＋1＝で1という結果がでてしまいます。これは数値も命令も，全部数字で書いてあるからこんなことになってしまうんです。つまり，結果が全然違ったものになってしまうのです。結果が違うくらいならまだいいのでして，このプログラム，2番地から実行を始めるととんでもないことになってしまいます。2番地には0があって，これはジャンプ命令でしたね。だから23番地へジャンプすることになります。23番地には，どんなデータが入っているのかさっぱりわかりません。だから，そこのデータによっては，あの，暴走を起こすかもしれないのです。

また，上の例で打ち込みを誤って3番地に0を入れてしまったら，0番地から実行したとしても，途中で(3番地のところで)31番地へジャンプします。

だから，マシン語の打ち込みは難しいのです。多分，その苦労を味わった方も多いことでしょう。そして，皆さんの苦労はこういうメカニズムから生じたのです。

マシン語のリストは 数字じゃない

以上説明してきたように，マシン語は全て数字で構成されています。が，実際にはマシン語のリストにAやらBやらのアルファベットも混じっています。話が違うじゃないか。いや，ごもっともで。

図2　16進数の数字

10進数	16進数	10進数	16進数
0	0	8	8
1	1	9	9
2	2	10	A
3	3	11	B
4	4	12	C
5	5	13	D
6	6	14	E
7	7	15	F

僕たちが普段使っている数は10進数といい，0から9の数字で構成され，9の次は繰り上がって10となるものです。最も，これに異論をとなえる人もいるんですが，ここでは当然省略します。詳しく知りたい方は，筒井康隆著「脱走と追跡のサンバ」を参考にしてください。文庫版も出ています。

という話は，とくに関係なく，筆者の好みから書いただけで，どうも失礼しました。

マイコンにおいては，よく16進数というのを使っているんです。10進数だと10になると繰り上がるのですが16進数では16で繰り上がります。10から15までを1桁で表現しなければならないのです。そこで，10から15をアルファベットのAからFに当てたわけです。マイコンのマニュアル（F-BASIC入門など）にもあるとおり，**図2**のような数字が10進数の0から15に相当するようになります。

で，なぜ16進数を使うかと言えば，コンピュータが2進法で動いているためなんです。**図1**を見てください。左の方の4桁の数がアドレス，その右の方に並んでいるのがデータです。全て16進数で記述されていますね。また，アドレスが4桁，データが2桁で記述されていることに気がついたでしょうか。

アドレスは4桁 データは2桁

16進数で2桁というのは，00〜FF まで表せ，10進数でいえば0から255までの数が表せます。0から255まで，つまり256とおりというのは2^8（2の8乗）なんですね。この8という数字，非常に意味があって，というのはマイコンが8ビットだからなのです。前に書いた，マシン語の説明を思い出してください。1つの数字で区切っていましたね。例えば1＋1の計算だと，

3　1　1　3　1　2

としてきました。ここでは区切りを1桁にして考えていたわけです。1桁，それも10進数だから0から9までの10個の数字にしていたんですね。ところが実際のコンピュータでは0から255，16進数では0からFF，2進数では00000000から11111111までを1区切りとして考えています。またアドレスは16進数4桁，つまり10進数の0から65535までを表します。0番地から65535番地までの記憶ができるということですね。

ねえ ねえ アセンブラって知ってる？

コンピュータの命令が数字で書かれていることは，わかっていただけたと思います。でも，それでは，どうあがいても人間に対して不親切にすぎますよね。この命令は1，あの命令は2……ではプログラムを作る前に，心身症か通り魔になってしまうことでしょう。でもプログラマーが通り魔になったとか，社会人としてやっていけなくなったとかいう話は僕の囲りの二，三の例を除けば，ないようです。と言うことは何かうまい方法が，つまり人間にも覚えられるような形でプログラムを組む方法がありそうです。で，アセンブリ言語と呼んでいます。

これは，命令一つ一つの意味を略して命令語を作ってしまうという考えからできたものなのです。例えば，たし算ならADD(Add)，引き算ならSUB(Substract)，JMP(Jump)，等々で，少なくとも数字よりは扱いやすくなっています。つまり1区切りの数は命令語とほぼ1対1の対応をしているといえます。このようなアセンブリ言語は，アセンブラと呼ばれるプログラムによって，マシン語すなわち数字に変換されるのです。

それでは，実際にアセンブリ言語をマシン語に変換したリストを見てみましょう。このリストは富士通から出ているアセンブラを使って出力したものです（**図3**）。

アセンブラのリスト

図3で一番左の数字は，行番号ですから無視してください。次の右の数字がアドレスです。16進数の4桁で表現されていますね。その右の2桁の16進数が，データとなっています。このデータが一つしかない行もあれば（たとえば200行），三つある行もあります（240行，280行など）。これは命令語の関係で，1命令が一つのデータになるものもあれば，五つものデータになるものもあるからです。

右側のアルファベットで書かれている部分がアセンブリ言語の部分で，これを読めばプログラムの流れや動作が詳細にわかるわけです。ま，もっともアセンブリ言語を知っていればの話ですけれども。

この稿は，このアセンブリ言語について詳しく述べるものではありません。この辺の詳細については別稿または文献を参照してください。ここの目的は，マシン語が入力できるようになることなんですよ。

というわけで，このリストの入力の仕方です。前に述べたとおり，左端がアドレスで，その右のいくつかの数字がデータですから，入力の際には**図4**のとおりに読めればよいことになります。

なお，290行，370行，400行，は，データの右側に入力してはいけない数字が書いて

図3　アセンブラ　ソースリスト

```
PAGE  001 (       ,000436) SORT

00100                                  NAM     SORT
00110                          ********************************
00120                          *      SORT PROGRAM v0.3        *
00130                          ********************************
00140                                  OPT     NOM
00150  3000                            ORG     $3000
00160  3000 34    76                   PSHS    X,D,U,Y
00170  3002 AE    02                   LDX     2,X       ;load DATA TOP ADDRESS-2
00180  3004 34    10                   PSHS    X
00190  3006 EC    81                   LDD     ,X++
00200  3008 58                         ASLB
00210  3009 49                         ROLA
00220  300A E3    E1                   ADDD    ,S++      ;DATA END ->D
00230  300C 34    06                   PSHS    D
00240  300E C3    0002                 ADDD    #2        ;DATA END+2 ->D
00250  3011 34    06                   PSHS    D
00260  3013 EC    84           XLOOP   LDD     ,X
00270  3015 31    02                   LEAY    2,X
00280  3017 10A3  A1           YLOOP   CMPD    ,Y++
00290  301A 2F    06     3022          BLE     NSWAP
00300                          *
00310                          *  SWAP D,(Y)
00320                          *
00330  301C EE    3E                   LDU     -2,Y
00340  301E 1E    03                   EXG     D,U
00350  3020 EF    3E                   STU     -2,Y
00360  3022 10AC  E4           NSWAP   CMPY    ,S
00370  3025 26    F0     3017          BNE     YLOOP
00380  3027 ED    81                   STD     ,X++
00390  3029 AC    62                   CMPX    2,S
00400  302B 26    E6     3013          BNE     XLOOP
00410  302D 32    64                   LEAS    4,S
00420  302F 35    76                   PULS    X,D,Y,U
00430  3031 39                         RTS
00440                                  END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=3000
PROGRAM END   ADDR=3031
PROGRAM ENTRY ADDR=****
```

図4　図3のリストのアドレスとデータ(16進数)

アドレス	3000	3001	3002	3003	3004	3005	3006	3007	……
データ	34	76	AE	02	34	10	EC	81	……

あります。注意してください。これはジャンプ先の番地で，290行では3022となっています。

アセンブラはメーカーによって若干の差異があります。が，ほとんど同じなので，どのアセンブリ言語のリストでも今回と同様の読み方ができると言えます。また，**図3**のようなリストを普通「ソースリスト(source list)」と言い，このリストからマシン語を入力する際にはアドレスに気をつけて入力することを心がけてください。データを入力するときに，確かにそのアドレスに入力しているかを見ていれば，リストの読み間違いは防げるでしょう。

16進数フェスティバル

上で紹介した，アセンブリ言語のリスト，皆さんが想像されるマシン語のリストとはかなり違ったものではなかったでしょうか。マシン語のリストと言えば，やっぱし，**図1**や**図5**のようなリストですよね。このリスト，見ていただけばおわかりのとおり，全て16進数で書いてあります。この**図5**は，**図3**のリストを記憶させたものと同じなのです。

さて，読み方ですが，一番左の4桁がアドレス，その右側から2桁の16進数が8つ並んでいるもの，これがデータです。こういったリストでは，一番左にあるアドレスが，そのすぐ右側にあるデータのアドレスを示しています。よって3000番地が34，3001番地が76，3002番地がAE…となるんですね。

言い忘れたことがありました。突然ですが，ここでちょっと触れておきたいと思います。というのは16進数のことで，3000と書いたときに10進数なのか16進数なのか区別がつかないという問題が出てきたからです。こういったとき，一般には，3000とかけば10進数の三千（つまり僕たちが普段使っている3000だね）で頭に＄をつけ＄3000とすれば16進数の3000を表しています。この16進数の3000，読み方は普通「さん，まる，まる，まる」とか「さん，ゼロ，ゼロ，ゼロ」などが使われます。ただ，人によっては「さんぜん」とも言っているようですが。こういった読み方では，16進数のAAAAを「えーせん，えーひゃく，えーじゅう，えー」とするんですよ。もっとも筆者はこの読み方，使ってませんけど。

それから＄3000のほかに，3000Hとかと&H3000，3000(16)とするやり方もあります。

これ以降，16進数と断わらなくても，数は全て16進で表します。

と，話を元に戻すと，**図5**の読み方でしたね。アドレスとデータ，この読み方はわかったと思います。で問題なのが，一番右側にある2桁の16進数なんです。このデータ，実はチェックサムと呼ばれるもので，入力してはいけないっ！データで，では，なぜついているかというと，だから，それはどうしてかというと，つまり，以下に説明します。

チェックサムって，チェックのためのサムなのさ

チェックサムとは，英語でcheck sumでチェックするサム（合計・総和）という意味を持っています。しかして，これはデータ入力の際の間違いを検出するためにあるのです。

では，どうやって間違いを見つけるのでしょうか。10進数を例に考えてみましょう。

データが，3と8と5だったとします。この3つの合計は16になりますよね。この下一桁の6を検出用に使ってみましょう。

もし，5と2を間違って入力したとすると，

3＋8＋2＝13

ここで下一桁は3となり入力を間違えたことがわかります。つまり，このチェックサムを何個かのデータに一つ付けると，チェックの手間が省かれることになります。例えば8個のデータに一つチェックサムを付ければ，データが256個あるときでも，チェックするものは32個しかありません。256個全部チェックするよりは，32個チェックした方が楽ですよね。

が，チェックサムには大きな欠点があります。

- 3＋8＋5＝16（チェックサム＝6）
- 3＋8＋2＝13（チェックサム＝3）
- 3＋5＋5＝13（チェックサム＝3）
- 3＋6＋7＝16（チェックサム＝6）

上がその例です。2番めの例と3番めの例では，チェックサムが同じようにおかしいのですが，間違っているところが違います。また4番めの例では，チェックサムは確かに正しいのですが，データは，これも確かに間違っています。まとめると，

① 間違っている場所を確定できない。
② チェックサムが正しくても，データが正しく入力されたとは言えない。

の2点となるのです。

この二つを解決するためには，データ一つにチェックサム一つをつければよいのです。が，これでは何のためのチェックサムかわかりません。つまり，なるべくたくさんチェックサムをつければいいのですが，それではチェックする項目が多くて疲れてしまいますよね。と言って，手抜きをしてチェックサムを減らせば（例えば，データ32個に一つ，データ128個に一つ，など），②のようなことの起こる確率が高くなってくるため，まずいんです。

そこで，妥当な線として，**図5**のように8つのデータに一つのチェックサムとすれば，それなりに効果があり，使えるものなのです。

また，チェックサムは16進のたし算で計算され，その下二桁を使っています。この計算は人間がやったのでは難しく面倒なのでコンピュータにやらせます。今月号の次の記事にチェックサムを計算する入力プログラムがありますから，Oh！FMではそれを使ってください。

図5　図3のマシン語リスト

```
Add   +0 +1 +2 +3 +4 +5 +6 +7  Sum
3000  34 76 AE 02 34 10 EC 81 :0B
3008  58 49 E3 E1 34 06 C3 00 :62
3010  02 34 06 EC 84 31 02 10 :EF
3018  A3 A1 2F 06 EE 3E 1E 03 :C6
3020  EF 3E 10 AC E4 26 F0 ED :D0
3028  81 AC 62 26 E6 32 64 35 :66
3030  76 39 00 00 00 00 00 00 :AF
```

これでおしまい

終わりました。

マシン語の原理とリストの見方を説明してきました。ご理解いただけましたでしょうか。まだまだお知りになりたいことが多いかと思います。とにかく何でも紹介していくつもりです。初心者の友達，バトミントン好きの田奈くんをよろしく。感想など，待ってます。

機械語とBASICの関連について

機械語とBASICの実行で大きな違いは，BASICは実行時に文法の誤りをチェックしているのに対し，機械語はそれを一切行わないことです。そのためBASICを止めるには BREAK キーを押せばよいのですが，機械語はリセットをかける（本体の向かって右端の白いボタンを押す）—— もちろんメモリの内容は空になります —— か，BREAK キーを押しながらリセットキーを押して先に離す —— メモリの内容は残っています —— の2種類の方法があります。前者をコールドスタートと言い，後者をホットスタートと呼んでいます。ホットスタートはウォームスタートとも呼ばれています。

CLEAR命令について

FM-7/8ともRAM領域は$0000～$7FFFの32Kバイトありますが，先頭の2Kバイト程度はF-BASICの作業領域として割り当てられているために，実質使用できるのは30Kバイトです。しかもDISK BASIC起動時は設定するファイルの数にもよりますが，大体DISKを処理する部分が増設（$7000～$7FFF）されているために使用可能な領域は21Kバイト～25Kバイトと狭くなります。もしこのメモリ領域に機械語のプログラムが存在していると，F-BASICが機械語を書き換える可能性が生じます。

そのようなことのないように場所を確保するのがCLEAR命令です。BASICの使用するトップアドレスは，$0041，0042番地に収納されているためこの値を参考にして決定すると良いでしょう。機械語を入力し，BASICと共存させる場合，CLEARを実行した方が安全です。

EXEC命令について

BASICからRAM上のサブルーチンをコールする命令です。実行時にはすべてのレジスタを保持しておくために自由にレジスタの破壊が可能です（ただしシステムスタックは別です。これを壊わすと復帰できません）。一度EXEC命令を実行するか，機械語ファイルをロードした後はEXEC⏎のみで実行可能です。

LOADM，SAVEMについて

機械語のセーブを行うのは，

◎機械語開始アドレス
◎機械語終了アドレス
◎機械語入口アドレス

の三種が必要です。ロードした時に入口アドレスがEXEC命令にセットされます。もし，データのファイルで入口アドレスが不必要でもやはり指定しないとエラーになります。ロード時には，

◎オフセット値
◎R（自動開始）

の二つを指定できますが，オプションなので，するしないは選択できます。

オフセットは全アドレスに可能なため，自由な番地にロードが可能となります。例えば二つの機械語を同じ番地で開発し，実行する時にずらしてロードし実行することが容易に行え，プログラムの管理上，非常に好都合です。

PEEK命令

メモリの内容を変数に引き渡す命令で，アドレスは0～65535，データは0～255で指定します。ただしアドレスがこの範囲をとれるのは，アドレスを指定している変数が実数のときのみで，整数の場合は$0000～$FFFFを0～32767，-32768～-1という数で表します。これを2の補数表現といい，整数変数は処理が複雑になるため，実数を使った方が問題がありません。機械語で処理した大量の結果をBASICが受け取るときに必要なコマンドです。

POKE命令

変数または定数の値を任意のアドレスに引き渡す命令で，アドレスに関する注意事項はPEEK命令と一緒です。機械語の量が少ない場合や，データ文に機械語を収納してロード時の手間を省いたり，大量のデータ（3Dグラフィックスなど）をBASICから機械語に自由な位置で引き渡しを行いたいときなどに使います。ただしアドレス管理をちゃんと行わないとBASICの作業領域等を書き換え，暴走を招くことがありますので注意が必要です。

DEFUSR～USRコールについて

引き渡す変数は一つですが，しばしば使われる機械語ルーチンは10個までならUSR関数で設定できます。この場合，レジスタはEXEC命令と異なり保持されないために，機械語の方で管理しないと暴走を招きます。変数の受け渡し方法はマニュアルを参照してください。USR関数を使用するときは必ず変数を受け取らないときでも引数を持たないとエラーになります。また変数は直接引き渡されるのではなく，アドレスの形で受け渡しが行われることに留意してください。

マシン語に関係するコマンドの形式

CLEAR 〔文字領域の大きさ〕〔，BASICの使用する上限のメモリアドレス〕
EXEC 〔開始番地〕
LOADM "ファイルディスクリプタ"〔,〔オフセット値〕〔,R〕〕
SAVEM "ファイルディスクリプタ"，開始番地，終了番地，入口番地
PEEK（式）
POKE 書き込み番地，データ
DEF USR〔数字〕＝整数
USR〔数字〕（引数）

マシン語特集

Oh!FM用チェックサムプログラム

本GO　誠司

マシン語のプログラムを作ったり入力するには，しっかりしたDOSなどがないと手入力によらざるを得ず，それはもう大変な作業です。まあ，人によっては,「僕は2Kバイトを1時間で入力できる！」などと逆にマシン語入力に命をかけている様な人もいますが，やはり大変な作業であることには違いありません。しかも,マシン語の場合,たった1バイトの入力ミスでもそれまでの苦労が水の泡となりかねません。そこで，BASICによる，簡単なマシン語入力，チェックサム付きダンププログラムを作ってみました。

チェックサムは，ご承知の様に，あるアドレスから何バイトかの和にアドレスを加えたものの下位1バイトで表現されています。Oh！FMでは，チェックサムを以下の様に統一しました。

1)$×××0～$×××7または$×××8～$×××Fの8バイトを対象とする。

2)1)の8バイトのメモリ内容の和の，下位1バイトをチェックサムとする。

ここで，アドレスを加えなかったのは，6809ではポジションインデペンデントなプログラムが多いので，アドレスを加えることはあまり意味を持たないと思われるからです。また，8バイトとしたことも,FMシリーズのモニタが8バイト単位で表示されるためです。以上に関して，いろいろとご意見もあるでしょうが，一応，Oh！FMでは上記の様に統一し，掲載のダンプリストなどもその様にするつもりです。

さて，プログラムの説明に移ります。まずリストを間違いなく打ちこんでセーブしてください。次にRUNさせると，

"D)ump or I)nsert ?"

と画面に出ますので，DまたはIの1文字を入力してください。Dではメモリのダンプとチェックサムを表示します。IではDプラスメモリチェンジ機能となります。どちらかのキー入力のあと，

"START ADDRESS =?"

と聞いてきますので，16進で4桁までの数値を入力してください。ただしAからFまでは大文字ですので注意してください。更に，

"END ADDRESS =?"

と出ますので，同様に入力してください。このとき，START ADDRESSより小さい

チェックサム　プログラムリスト


```
100 WIDTH 40,25:CONSOLE 4,21,0,0:COLOR 7
110 DIM CSUM(8,15)
120 LOCATE 0,0:INPUT "D)ump or I)nsert :";A$:CLS
130 LOCATE 0,0:INPUT "START ADDRESS =";ADS$:LOCATE 15,0:PRINT "&H";ADS$
140 INPUT "END   ADDRESS =";ADE$:LOCATE 15,1:PRINT "&H";ADE$
150 ADS=VAL("&H"+ADS$):ADE=VAL("&H"+ADE$)
160 PRINT "Add   +0 +1 +2 +3 +4 +5 +6 +7  Sum";
170 COLOR 4:LOCATE 0,4
180 IF A$="I" OR A$="i" THEN 270
190 ' DUMP MAIN
200 FOR H=ADS TO ADE STEP 128
210  GOSUB400
220  IF INKEY$="" THEN 220
230  CLS 1
240 NEXT H
250 WIDTH 40,25:END
260 ' INSERT MAIN
270 FOR H=ADS TO ADE STEP 128
280 DX=6:DY=4:GOSUB 400
290 GOSUB 600
300 CLS 1
310 NEXT H
320 WIDTH 40,25:END
400 'DUMP ROUTINE
410  FOR I=H TO H+127 STEP 8
420   P$=STRING$(4-LEN(HEX$(I)),"0")+HEX$(I)
430   PRINT P$;"  ";
440   SUM=0
```

```
450    FOR J=0 TO 7
460     A$=RIGHT$("0"+HEX$(PEEK(I+J)),2)
470     SUM=SUM+PEEK(I+J):CSUM(J,(I-H)¥8)=PEEK(I+J)
480     PRINT A$;" ";
490    NEXT J
500    PRINT ":";RIGHT$("0"+HEX$(SUM),2)
510   NEXT I
520 PRINT
530 RETURN
600 'INKEY
610 I$=INKEY$
620 IF I$="" THEN GOSUB810 :GOTO 610
630 IF (I$<"0" AND (I$<>CHR$(13) AND I$<>CHR$(28) AND I$<>CHR$(29) AND I$<>CHR$(
32))) OR (I$>"9" AND I$<"A") OR I$>"F" THEN BEEP:GOTO 610
640 IF I$=CHR$(29) THEN GOSUB 770:GOTO 610
650 IF I$=CHR$(28) OR I$=CHR$(13) THEN GOSUB 730:GOTO 610
660 IF I$=CHR$(32) THEN RETURN
670 LOCATE DX,DY:PRINTI$;:ADD=H+(DX-6)¥3+(DY-4)*8
680 IF DX MOD 3=0 THEN DAA=16*VAL("&H"+I$)+PEEK(ADD) MOD 16
690 IF DX MOD 3=1 THEN DAA=(PEEK(ADD)¥16)*16+VAL("&H"+I$)
700 POKE ADD,DAA:CSUM((DX-6)¥3,(DY-4))=DAA
710 DX=DX+1:IF (DX-8) MOD 3=0 THEN GOSUB 730
720 GOTO 610
730 ' FOWARD
740 DX=((DX-6)¥3)*3+9
750 IF DX>27 AND DY<19 THEN GOSUB 870:DX=6:DY=DY+1 ELSE IF DX>27 AND DY=19 THEN
GOSUB 870:DX=27:BEEP
760 RETURN
770 ' BACK
780 DX=((DX-6)¥3)*3+3
790 IF DX<6 AND DY>4 THEN GOSUB 870:DX=27:DY=DY-1 ELSE IF DX<6 AND DY=4 THEN DX=
6:BEEP
800 RETURN
810 ' CURSOR
820 FOR Z=0 TO 1
830  LINE(DX*16,DY*8)-(DX*16+15,DY*8+7),XOR,4,BF
840  FOR TI=0 TO 50:NEXTTI
850 NEXTZ
860 RETURN
870 ' CHECKSUM
880 SUM=0:FOR S=0 TO 7:SUM=SUM+CSUM(S,DY-4):NEXT S:LOCATE 31,DY:PRINT RIGHT$("0"
+HEX$(SUM),2)
890 RETURN
```

数字を入力すると，128バイトのみ表示して終わってしまいます。ここまで入力し終わるとおもむろにダンプを始めます。128バイト表示すると，Dumpの場合はフリーズ状態となり，何かkeyを押すまでそのままですので，じっくりとチェックサムなどを確認してください。Insertの場合は左上にカーソルが点滅しますので，16進数を2桁でゆっくりと入力すればメモリ内容の変更が可能です。

この場合もAからFまでは大文字ですので間違いのない様に！　2桁入力すると自動的にアドレスが一つ増え，カーソルは隣に移ります。ただ単にカーソルを移動したいときは，"←"と"→"のカーソル移動key を押してください。1バイト単位でカーソルが移動しますので，間違って入力したところも簡単に修整できます。今回は「なるべく簡単なプログラムで！」とのことなので，1バイト入力するのに必ず2桁の数値を入力するようになっています。例えば"A 0"と入力すべきところを"A 1"と入力してしまった場合，カーソルを移動させると"A"のところにきてしまい，"1"のところへはダイレクトにきません。ですからこの様な場合，再度"A 0"と入力し直してください。ただ，ちょうどカーソルが移動したところのみ直したければ，再入力後カーソル移動かCRで十分です。1行分の入力，またはカーソル移動が終わると，チェックサムは自動的に計算されますので8バイト入力するごとに確認ができます。この様に入力，変更をしていき，次の128バイトへ進む場合には，スペースバーを押してください。以下同様にしてマシン語を入力していき，初めに入力したエンドアドレスまで続きます。もし途中でやめたい時にはSTOP キーで抜け出せます。ただし，この時にはコンソールを直す必要がありますが……。

さて，以上の様にしてめでたくマシン語の入力も終わり，チェックサムも全て正しいぞ！　とばかりにいきなりEXECなどとはしないでください。必ずセーブをしてからEXECしましょう！　これだけはいつも念頭においてプログラムを組みましょう。また，ディスクをつないでいる人は，EXECの時にドアをオープンにして，暴走によるディスク内容の破壊に備えてください。かつて，自分で組んだマシン語プログラムを実行させたときに（デバックもせずいきなり！），ディスクが作動して，みごとにデータを破壊してしまった苦い経験が私にはあります。以来，必ずドアを開けてからEXECすることが癖となり，データの破壊もなくなりました。まあ，Oh！FMの読者の皆さんはこんなことないでしょうが……。

なお，FM-11の場合はSCREEN 0を最初に実行してください。

マシン語特集

マシン語プログラム実例集 1

演算のアルゴリズムとプログラム例

鶴岡　哲明

創刊号，第２号と２回にわたりマシン語入門を書いてきたが，実例が少なく，これだけでは自分の望む処理をどう行ったらよいかわからない，またアドレッシングをどう使ったらよいかわからないという人が多いと思う。そこで，今回はマシン語プログラムの実例をいくつか挙げて，命令表を見ただけではわからない種々の処理(割算，平方根，多倍長の計算)などについて触れてみよう。
また，マシン語プログラミングに欠かせないアセンブリ言語の表記上の注意などにも触れたいと思う。

マシン語には，加算・減算の命令があり，また，８ビット以上の精度を要する計算もそれぞれ繰り上がり・繰り下がりを含めての加・減算命令があるので，とくに頭を使わなくても大丈夫である。しかし，乗算については，この命令だけでは多倍長（２バイト以上で表される数値）の乗算はできない。また，除算になると除算命令すらない。この章では，これらの計算法（多倍長も含む）を紹介していこう。

また，BASIC では種々の数値関数が用意されているが，マシン語ではこれらをどう処理したらよいかについても述べてみたい。

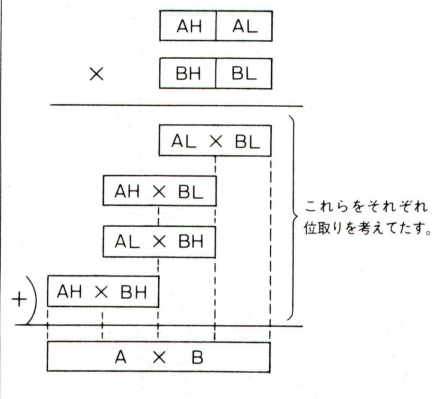
乗　　算

6809 CPU は８ビット同士の乗算なら一つの命令で済む。それでは16ビット以上の場合はどうしたらよいのか。基本的には，我々が筆算で行うのと同じことをプログラミングしてやればよい。我々が筆算で乗算をする場合，10進数で１桁ずつ計算し，それぞれを加えていく。このとき，10進数１桁同士の乗算は九九を覚えて行う。つまり１桁同士の乗算だけ覚えておき，あとは，これと加算を組み合わせて行うのである。

これがマイコンになると，6809の場合，MUL 命令があるため，一度に８ビット（２進数で８桁，16進数では２桁，256進数で１桁）単位で計算できる。つまり，人間が計算するやり方で，１桁が８ビット分あると考えて計算法をコーディングすればよい。具体的には**図１**のような処理をすればよいのである。この図は16ビット同士の場合だが，ビット数が増えても同じようにする。例えば32ビット（＝４バイト）同士なら４×４＝16回の乗算をすればよいことになる。

図１　16ビット符号なし同士の乗算

A×B
AH：Aの上位８ビット(上位バイト)
BH：Bの上位８ビット(上位バイト)
AL：Aの下位８ビット(下位バイト)
BL：Bの下位８ビット(下位バイト)

```
              | AH | AL |
    ×         | BH | BL |
  ---------------------------
              | AL × BL |  ┐
         | AH × BL |       │
         | AL × BH |       │ これらをそれぞれ
 +) | AH × BH |            ┘ 位取りを考えてたす。
  ---------------------------
    |      A  ×  B       |
```

さて，**図１**をアセンブリ言語でプログラムしたものが**リスト１**である。これは富士通のアブソリュートアセンブラを用いてアセンブルした。ここでアセンブリ言語について少々解説する。

アセンブリ言語は，マシン語と基本的には一対一で対応するものであることは言うまでもない。通常，マシン語でプログラムを書くとき，直接マシンコード（16進数）で書く人はまずいないだろう。たとえアセンブラを使わなくても，まずニモニック(アルファベット３～５文字)と必要に応じてオペランド（操作の対象）を記号で表記する。つまり，アキュムレータAに０を入れる命令を

　　LDA　＃０

と表記する。これを命令表でマシンコードに直すわけだ。

この作業をコンピュータ自体にやらせるのがアセンブラであることは言うまでもないが，アセンブラを使う場合，種々の決まりがある。**リスト１**を元に解説しよう。

NAM……これはプログラムの頭に付け，プログラムの名称を示す疑似命令（アセンブラに対する命令）である。おまじないにすぎないが，これをつけなくてはいけないアセンブラが多い。アセンブルリストの頭に名を出してくれる

リスト 1

```
PAGE  001 (       ,034038) WMUL

00010                               NAM   WMUL
00020                               OPT   O
00030  3000                         ORG   $3000
00040  3000 34   16                 PSHS  X,D
00050  3002 AE   02                 LDX   2,X
00060  3004 8D   03   3009          BSR   WMUL
00070  3006 35   16                 PULS  X,D
00080  3008 39                      RTS
00200            3009       WMUL    EQU   *
00210  3009 CC   0000               LDD   #0
00220  300C ED   04                 STD   4,X
00230  300E A6   01                 LDA   1,X
00240  3010 E6   03                 LDB   3,X
00250  3012 3D                      MUL
00260  3013 ED   06                 STD   6,X
00270  3015 A6   01                 LDA   1,X
00280  3017 E6   02                 LDB   2,X
00290  3019 3D                      MUL
00300  301A E3   05                 ADDD  5,X
00310  301C ED   05                 STD   5,X
00320  301E 86   00                 LDA   #0
00330  3020 89   00                 ADCA  #0
00340  3022 A7   04                 STA   4,X
00350  3024 A6   84                 LDA   ,X
00360  3026 E6   03                 LDB   3,X
00370  3028 3D                      MUL
00380  3029 E3   05                 ADDD  5,X
00390  302B ED   05                 STD   5,X
00400  302D A6   04                 LDA   4,X
00410  302F 89   00                 ADCA  #0
00420  3031 A7   04                 STA   4,X
00430  3033 A6   84                 LDA   ,X
00440  3035 E6   02                 LDB   2,X
00450  3037 3D                      MUL
00460  3038 E3   04                 ADDD  4,X
00470  303A ED   04                 STD   4,X
00480  303C 39                      RTS
00490                               END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=3000
PROGRAM END   ADDR=303C
PROGRAM ENTRY ADDR=****
```

ので便利。

OPT……これも疑似命令の一つで，アセンブル時のさまざまな指定（アセンブルリストの出力など）を行うもので，プログラムを見るときには枕詞だと思って指で隠してほしい。

ORG……この命令のあとのプログラムが何番地から始まるかを示す。

EQU……ラベルの値を設定する。EQUのあとの*はこのアドレスを示す。つまり，**リスト1**の行番号200の場合は，WMULというラベルは＄3009という数値を表す。

EQUの後には数値をつける場合も多い。また，ラベルの値の設定は，他に命令の前にラベルをつけて，その命令のアドレスをラベルが示す，という方法もある。この例は**リスト1**にはないが，後でたくさん出てくる。

END……アセンブル終了の命令

これらの命令は，アセンブラに対する命令で，直接CPUが実行するマシン語とは異なるため，疑似命令と呼ばれる。ほかにもいろいろあるが，出てきたときに解説しよう。

さて，アセンブルリストの見方を**図2**に挙げておく。ここでは，他に疑似命令RMBを使っている。

では**リスト1**の解説に移ろう。これは，Xレジスタの示すアドレスに入っている2バイトのデータと，Xレジスタの値＋2のアドレスの2バイトをかけてXレジスタの値＋4のアドレスからの4バイトに結果を入れるプログラムである。プログラム自体は行番号200から始まっている。行番号40から80までは，この乗算ルーチンテスト用のものである。

この例ではループを使っていない。それはこの程度の繰り返し回数ではループを使ってもプログラムはそう短くならず，時間だけかかってしまうからだ。しかし，3バイト同士の乗算以上になるとループを使うべきだろう。

なお，このプログラムは，先頭アドレスをどこにしてもそのまま走る。このアドレスでは不都合なら，そのまま移せばよい。

さて，せっかくの乗算プログラム，実際に試してみたいと思う。そこで，テスト用のプログラムを作ってみた。**リスト2**である。そして，このプログラムと先の**リスト1**の行番号200からのプログラムの間をとりもつのが**リスト1**の行番号80までなのである。

リスト2では，マシン語プログラムをUSR関数を用いて呼び出す。USR関数は，

図2

```
PAGE  001 (       ,000318) TEST

00010                               NAM   TEST
00020  2000                         ORG   $2000
00030  2000 86   00                 LDA   #$00
00040  2002 B7   2006               STA   LAB
00050  2005 39                      RTS
00060  2006      0001       LAB     RMB   1
00070                               END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=2000
PROGRAM END   ADDR=2006
PROGRAM ENTRY ADDR=****
```

リスト2

```
10 'WMUL TEST
20 DEFINT X
30 DIM X(3)
40 LOADM"WMUL"
50 DEFUSR=&H3000
60 DEFFNR(X)=INT(RND(1)*X)
70 X(0)=FNR(32767):X(1)=FNR(32767):PRINTX(0),X(1),:D#=CDBL(X(0))*CDBL(X(1)):PRIN
TD#,
80 A=USR(VARPTR(X(0))):B=X(3):IF B<0 THEN B=B+65536!
90 C#=X(2)*65536#+B:PRINTC#:IF C#<>D# THEN BEEP
100 GOTO 70
```

Xレジスタの示すアドレスから引数を入れ，アキュームレータAに数値の型を入れてDEFUSRで指定したアドレスを呼び出すことは，FMシリーズに付属のマニュアルに書いてあるが，実数型の取り扱いは面倒なので，今回はすべて整数型で用いることにした。また，USR関数では一度に数値を一つしか渡せない。そこで，配列を用いることにした。配列の変数は，メモリに**図3**のように格納されている。そして，この配列の先頭アドレスは，VARPTR関数を用いてやればよい。例えば，

VARPTR(X(0))

は，X(0)のアドレスとなる。これをもとに，X(1)，X(2)，……のアドレスもわかる。**リスト2**のプログラムでは，整数型の配列X(3)を宣言し，**リスト1**のプログラムをUSR関数で呼ぶ。このとき，Xレジスタの値＋2のアドレスに，X(0)のアドレスが入っているので，この値をXレジスタに入れる。この処理が**リスト1**の行番号50である。そしてWMULをサブルーチンコールすると，Xレジスタの示すアドレスに入っている2バイトとその次の2バイトを掛けて，その次の4バイトに結果を入れるので，BASIC側からみると，X(0)とX(1)を掛けた値の上位2バイトがX(2)に，下位2バイトがX(3)に入ることになる。

ここで注意しなくてはならないのは，WMULはすべて符号なしで処理するので，整数の−32768〜−1を32768〜65535と判断する。そのため，**リスト2**のプログラムではそれぞれに32767までの数しか入れていない。また結果も符号なしなので，X(3)が下位16バイトとなるには，X(3)が負の場合は65536を加えてやらなくてはならない。X(2)の方は，結果が先の制限のために大きくならず，補正するまでもない。

さて，**リスト2**のプログラムを実行すると，BASICの倍精度で計算した結果とWMULで計算した結果が並んで出る。当然ぴたりと一致する。しかし，BASICで単精度で計算すると結果は合わない（単精度では誤差が出てしまう）。

なお，メモリ上に**リスト1**のプログラムが入っていれば，**リスト2**のLOADM（行番号40）はいらない。

さて，このプログラムは符号なし乗算であったが，符号付きでやるように変更するには少し手間がかかる。この場合，結果の符号をあらかじめ調べ，掛け合わせる2数の絶対値をそれぞれとってWMULを呼び，結果を先に調べた正負によって，結果に符号を含めてやればよい（すなわち，負なら0から演算結果を引くことで，結果にマイナス1をかけたことになる）。

さて，この例ではMUL命令を使ったがMUL命令を使わなくても乗算はできる。乗算命令を持たないCPUでは，こちらの方式を使う。つまり，先ほどの例では一度に8ビットずつ掛けたが，1ビットずつ掛けるのである。1ビットだったら，0か1しかないので1倍と0倍には乗算はいらない。そして，それぞれの位の分だけASL，ROLを使ってシフトしてやればよい。この様子を**図4**に示した。この**図4**をマシン語で実現したのが**リスト3**である。行番号80までは**リスト1**と同じである。本体は行番号100のWMULSHからである。

変数エリアについても同じだが，こちらはワークエリアが余計にいるのと，かける2数の値が保存されない。さて，そのワークエリアであるが，このプログラムではスタックを使っている。スタックを使うとワークエリアをどこに設けるかで悩む必要はない。ただしスタックポインタの値をきちんと元に戻すことを忘れないようにしてほしい。

図3　配列の格納のされ方

図4　シフトによる乗算

リスト 3

```
PAGE   001 (       ,042306) WMULSH

00010                                   NAM    WMULSH
00020                                   OPT    0
00030  3000                             ORG    $3000
00040  3000 34   16                     PSHS   X,D
00050  3002 AE   02                     LDX    2,X
00060  3004 8D   03    3009             BSR    WMULSH
00070  3006 35   16                     PULS   X,D
00080  3008 39                          RTS
00100            3009         WMULSH    EQU    *
00110  3009 C6   10                     LDB    #16
00120  300B 34   04                     PSHS   B
00130  300D CC   0000                   LDD    #0
00140  3010 ED   04                     STD    4,X
00150  3012 ED   06                     STD    6,X
00160  3014 34   06                     PSHS   D
00170  3016 64   02           LOOP      LSR    2,X
00180  3018 66   03                     ROR    3,X
00190  301A 24   0E    302A             BCC    EADD
00200  301C EC   84                     LDD    ,X
00210  301E E3   06                     ADDD   6,X
00220  3020 ED   06                     STD    6,X
00230  3022 EC   E4                     LDD    ,S
00240  3024 E9   05                     ADCB   5,X
00250  3026 A9   04                     ADCA   4,X
00260  3028 ED   04                     STD    4,X
00270            302A         EADD      EQU    *
00280  302A 68   01                     ASL    1,X
00290  302C 69   84                     ROL    ,X
00300  302E 69   61                     ROL    1,S
00310  3030 69   E4                     ROL    ,S
00320  3032 6A   62                     DEC    2,S
00330  3034 26   E0    3016             BNE    LOOP
00340  3036 32   63                     LEAS   3,S
00350  3038 39                          RTS
00360                                   END
TOTAL  ERRORS 00000--00000
TOTAL  WARNINGS 00000--00000

PROGRAM BEGIN ADDR=3000
PROGRAM END   ADDR=3038
PROGRAM ENTRY ADDR=****
```

図5　WMULSHを呼び出したとき

さて，このプログラムがどういう処理をしているかというと，かける数（**図4**では34）を右シフトする。すると下位ビットがキャリーフラグに入る。そしてキャリーフラグが1なら，結果にかけられる数（**図4**では12）を加える。次にかけられる数を2倍（左シフト）し，またかける数（このときはすでに1ビットずれているが）を右シフトして…を計16回繰り返せば全ての位について計算が終了するわけである。なお，かけられる数を最終的に16回シフトするので，2バイトでは足らず，2バイトの追加上位バイトが必要である。また16回のカウント用のメモリも必要で，これら3バイトがワークエリアとなり，スタックを用いている。スタックの使われ方を**図5**に示した。

除　　算

さて，次は除算である。こちらはもうシフトを用いてやるしかない。方法は我々が筆算でやるのと同じ方法である。10進数でやるのをそのまま2進数に置き換えればよい。10進数でやる場合よりも引き算の回数が増えるが，2進数だと1が立つか0が立つかだけ考えればよいので，単純に比較で済む。10進数だと1～9のどれが立つか予想しなくてはならない。このため，アルゴリズム的にはずっとシンプルになる。この様子は**図6**を見てほしい。ただ，機械がやるのと人間がやるのとでは発想的に異なり，割

図6　2進数の割り算

り切れても割り切れたなどと考えるより強引に計算を続けた方がよいのである。余計な判断は時間がかかり，この程度のことならそのままやってしまった方が早い。

さて，**図6**のアルゴリズムをどう実現するかであるが，これも乗算と同様シフトを使えばよい。割られた数を左シフトし，飛び出てきた分と割る数を比べ，割る数が大きければ0を結果に入れ，そうでなければ1を結果に入れ，飛び出た部分から割る数を引く。16ビット÷16ビットなら，これを16回繰り返せば商が求められ，またあまりも出る。また，1の位が求められても，まだこの操作を繰り返せば，小数点以下も求められるがこの場合，2進数の小数となることに注意しなければならない。

これをコーディングしたものが**リスト4**

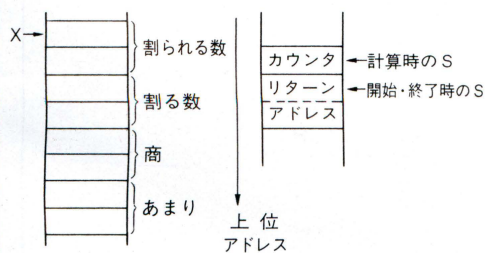
リスト 4

```
PAGE   001 (        ,040904) WDIV

00010                            NAM    WDIV
00020                            OPT    O
00030  3000                      ORG    $3000
00040  3000 34   16              PSHS   X,D
00050  3002 AE   02              LDX    2,X
00060  3004 8D   03   3009       BSR    WDIV
00070  3006 35   16              PULS   X,D
00080  3008 39                   RTS
00100            3009      WDIV  EQU    *
00110  3009 86   10              LDA    #16
00120  300B 34   02              PSHS   A
00130  300D CC   0000            LDD    #0
00140  3010 ED   04              STD    4,X
00150  3012 68   05        LOOP  ASL    5,X
00160  3014 69   04              ROL    4,X
00170  3016 68   01              ASL    1,X
00180  3018 69   84              ROL    ,X
00190  301A 59                   ROLB
00200  301B 49                   ROLA
00210  301C 10A3 02              CMPD   2,X
00220  301F 25   04   3025       BLO    AINC
00230  3021 A3   02              SUBD   2,X
00240  3023 6C   05              INC    5,X
00250  3025 6A   E4        AINC  DEC    ,S
00260  3027 26   E9   3012       BNE    LOOP
00270  3029 ED   06              STD    6,X
00280  302B 32   61              LEAS   1,S
00290  302D 39                   RTS
00300                            END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=3000
PROGRAM END   ADDR=302D
PROGRAM ENTRY ADDR=****
```

（00040〜00080：BASICとの間をとりもつ部分、00100〜00300：割り算部）

図7　リスト4のワークエリア

である。これも乗算と同様にXレジスタの値，Xレジスタの値＋2で示されるアドレスにそれぞれ割られる数，割る数を入れれば商とあまりがXレジスタの値＋4，Xレジスタの値＋6に入ってくる。

このプログラムで他に必要なワークエリアは16回のカウント用の1バイトだけであるが，これもスタックを使うことにした。また，割られる数は直接して，飛び出た分はDレジスタを使っている。ところでこのプログラムは割る数が0の時をチェックしていないが，前もって判断させれば済む。

こちらも乗算と同様，BASICで試せるようにした。言うまでもなくX(0)÷X(1)の商がX(2)に，あまりがX(3)に入る。あまりをさらにX(1)で割って，BASICのみと比較できるようにした（**リスト5**）。マシン語の方は当然符号なしで扱っているが，BASICの方で結果が符号なしでも符号付きでも同じになる範囲でしか使っていないので補正はしていない。

なお，符号付き割り算をしたい場合には，乗算同様，前もって2数の符号から結果の符号を判断し，絶対値で割り算をしてから結果を符号付きにすればよい。

また，もっと多倍長の除算については，それぞれの変数を大きく取ればよいが，今度はDレジスタでは収まらなくなるので，プログラムは大幅変更となる。

平　方　根

3番目は平方根である。平方根の計算は大変そうに見えるが，これが案外簡単にできる。筆算で平方根を計算するのは通常開平法を用いる。やり方は詳しくは中3の数学の教科書などに参考として載っていると思うが，簡単な解説を**図8**に挙げておく。(原理は省略させていただく)。

高校生以上の方は数列の極限でもできると言うであろうが，この方法は確実で速い(恐らくBASICのSQR関数もこの方法のはずだ)。

このやり方は2進数でも同じで，割り算と同じく，10進だといくつが立つかが10とおり考えられ，見当をつけてやるが，2進数だと0か1なので，1で試してみて，OKならそのまま，だめなら0にすればよいのではるかに楽である。

さて，具体的にはどうすればよいか。コーディング例を**リスト6**に挙げる。例によって行番号80まではBASICとの間をとりもつ部分で，本体は行番号100からである。さて，またアセンブリ言語の使い方で新しいものが出たので説明しよう。コメント(BASICのREM)である。これは，頭に＊をつければよい。また，命令の行で，オペランドのあとに区切り記号を入れて（またはある程

リスト 5

```
10  WDIV TEST
20 DEFINT X
30 DIM X(3)
40 LOADM"WDIV"
50 DEFUSR=&H3000
60 DEFFNR(X)=INT(RND(1)*X)
70 X(0)=FNR(32767):X(1)=FNR(32767):PRINTX(0),X(1),X(0)/X(1),
80 A=USR(VARPTR(X(0)))
90 PRINT X(2),X(3)/X(1)
100 GOTO 70
```

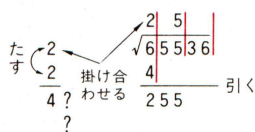



図8 開平法

【例：√65536】

・小数点の位置から2桁ずつ区切る。
・?×?が6を超えない?を探す。
?は2

・4?×?が255を超えない?を探す。
?は5

50?×?が3036を超えないような?を探す。
?は6

ここが0ならちょうど開けたことになる。
0でないときに，さらに続けると小数点以下まで求めることになる。
小数点以下を2桁ずつ送っていってやる。

【例：√2】

```
 1                1. 4  1  4……
 1              √2.000000
---              1
 24              ------
  4              1 00
---                96
 281              ------
   1              400
---               281
 2824             ------
    4            11900
---              11296
 2828            ------
                   604
```

度スペースを入れて）書くことができる。これは用いるアセンブラによって異なる場合があるので注意してほしい。

まず変数についてであるが，Xレジスタの示すアドレスに入った2バイトの整数の√￣を計算してXレジスタ+2のアドレスに入れる。結果は1バイトで表せるので，ついでに小数点以下も計算することにした。つまり，Xレジスタの値+3のアドレスには小数部の256倍が入る。

内部の説明の前に，BASICのテストプログラムを使って試してみよう。例によって配列を使う。すると，X(0)に平方根を求めたい数を入れればよい。しかし，今度は結果の見方が違う。**リスト6**のプログラムは結果の整数部がXレジスタの値+2のアドレスに，小数部の256倍がXレジスタの値+3のアドレスに入る。したがって，BASIC側でX(1)を読みだすとこれを符号なし整数とみるための操作(負なら65536を加える)をすれば結果の256倍となる。これを256で割れば結果が有効数字5桁弱で求められる。また，整数部を表す1バイト

リスト6

```
PAGE  001 (        ,003116) SQRT

00010                               NAM    SQRT
00020                               OPT    0
00030    3000                       ORG    $3000
00040    3000 34    16              PSHS   X,D
00050    3002 AE    02              LDX    2,X
00060    3004 8D    03    3009      BSR    SQRT
00070    3006 35    16              PULS   X,D
00080    3008 39                    RTS
00100               3009      SQRT  EQU    *
00110    3009 CC    0000            LDD    #0
00120                         * CLEAR WORK AREA(8 BYTEFROM S.TOP)
00130    300C 34    06              PSHS   D
00140    300E 34    06              PSHS   D
00150    3010 34    06              PSHS   D
00160    3012 34    06              PSHS   D
00170    3014 C6    10              LDB    #16       ;COUNTER SET
00180               3016      LOOP  EQU    *
00190                         * SHIFT PARAMETER
00200    3016 68    01              ASL    1,X
00210    3018 69    84              ROL    ,X
00220                         * SHIFT WORK1
00230    301A 69    63              ROL    3,S
00240    301C 69    62              ROL    2,S
00250    301E 69    61              ROL    1,S
00260    3020 69    E4              ROL    ,S
00270                         * SHIFT PARAMETER
00280    3022 68    01              ASL    1,X
00290    3024 69    84              ROL    ,X
00300                         * SHIFT WORK1
00310    3026 69    63              ROL    3,S
00320    3028 69    62              ROL    2,S
00330    302A 69    61              ROL    1,S
```

```
00340   302C 69    E4                  ROL    ,S
00350                           *
00360   302E 1A    01                  ORCC   #$01     ;SEC
00370                           * ROTATE WORK2
00380   3030 69    67                  ROL    7,S
00390   3032 69    66                  ROL    6,S
00400   3034 69    65                  ROL    5,S
00410   3036 69    64                  ROL    4,S
00420                           * COMPARE WORK1-WORK2
00430   3038 A6    63                  LDA    3,S
00440   303A A0    67                  SUBA   7,S
00450   303C A6    62                  LDA    2,S
00460   303E A2    66                  SBCA   6,S
00470   3040 A6    61                  LDA    1,S
00480   3042 A2    65                  SBCA   5,S
00490   3044 A6    E4                  LDA    ,S
00500   3046 A2    64                  SBCA   4,S
00510                           *
00520   3048 24    08    3052          BHS    SET
00530   304A 6A    67           CLRLSB DEC    7,S      ;CLEAR LSB OF WORK2
00540                           * SHIFT ANSWER
00550   304C 68    03                  ASL    3,X
00560   304E 69    02                  ROL    2,X
00570   3050 20    20    3072          BRA    DCT
00580              3052         SET    EQU    *
00590                           * WORK1-WORK2->WORK1
00600   3052 A6    63                  LDA    3,S
00610   3054 A0    67                  SUBA   7,S
00620   3056 A7    63                  STA    3,S
00630   3058 A6    62                  LDA    2,S
00640   305A A2    66                  SBCA   6,S
00650   305C A7    62                  STA    2,S
00660   305E A6    61                  LDA    1,S
00670   3060 A2    65                  SBCA   5,S
00680   3062 A7    61                  STA    1,S
00690   3064 A6    E4                  LDA    ,S
00700   3066 A2    64                  SBCA   4,S
00710   3068 A7    E4                  STA    ,S
00720   306A 6C    67                  INC    7,S      ;INC WORK2
00730   306C 1A    01                  ORCC   #$01     ;SEC
00740                           * SHIFT ANSWER
00750   306E 69    03                  ROL    3,X
00760   3070 69    02                  ROL    2,X
00770   3072 5A                 DCT    DECB
00780   3073 26    A1    3016          BNE    LOOP
00790   3075 32    68                  LEAS   8,S      ;WORK AREA DELETE
00800   3077 39                        RTS
00810                                  END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=3000
PROGRAM END   ADDR=3077
PROGRAM ENTRY ADDR=****
```

リスト 7

```
10 'SQRT TEST
20 DEFINT X
30 DIM X(1)
40 LOADM"SQRT"
50 DEFUSR=&H3000
60 DEFFNR(X)=INT(RND(1)*X)
70 X(0)=FNR(32767):PRINTX(0),SQR(X(0)),
80 A=USR(VARPTR(X(0))):B=X(1):IF B<0 THEN B=B+65536!
90 PRINT PEEK(VARPTR(X(1))),B/256
100 GOTO 70
```

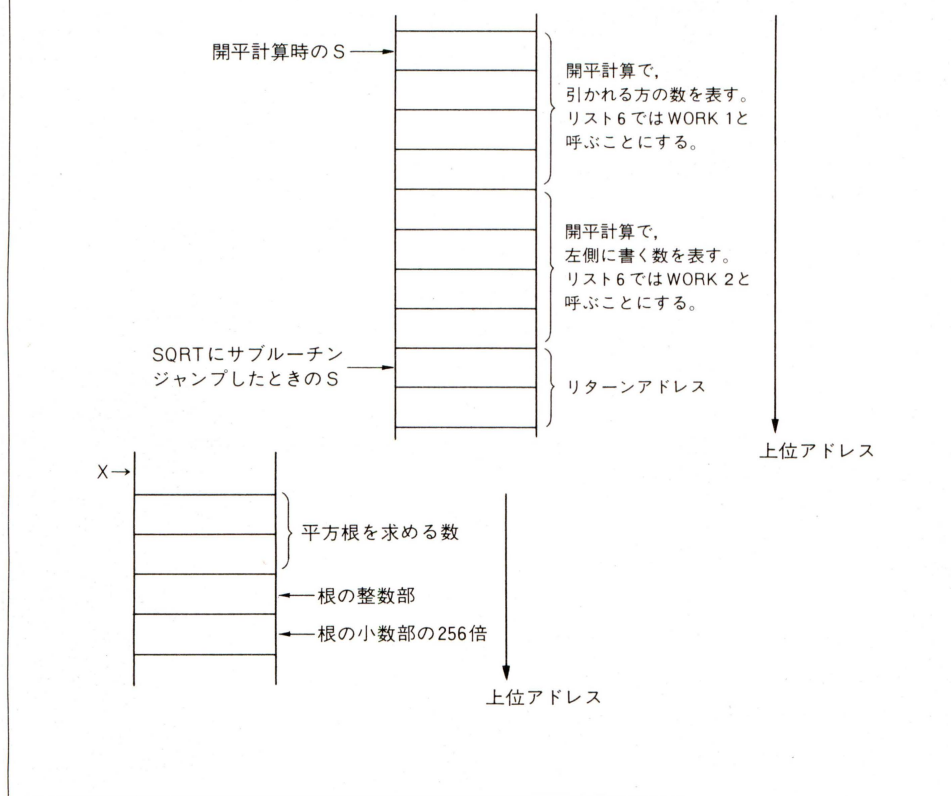
は，X(1)の上位バイトのアドレスに入っている。つまり，

PEEK(VARPTR(X(1)))

で求めてもよい。

以上の処理をして BASIC の SQR 関数と比べるプログラムが**リスト 7** である。

表示される数値も BASIC で計算された方は 6 桁程度に四捨五入しているためもあり，**リスト 6** のプログラムで計算したものとそれほど差がでない。

図9　リスト6のプログラムのワークエリア

では**リスト6**の解説に移ろう。

まずワークエリアについてであるが，これは**図9**のようになっている。これも必要なワークエリアにスタックを使っている。平方根を求める数を1回のループに付き2ビットシフトしていき，はみ出した部分をスタックポインタの値＋0～＋3に送る。また，スタックポインタの値＋4～＋7の方の最下位ビットを1に，あとは左にシフトする（これはキャリーフラグを1にしてROLを実行すればよい）。そしてこの2数を比べる。比べ方であるが多倍長の比較命令はないから，SBCを用いて，結果はメモリに格納しない方式をとる。そしてその大小に応じた処理をする。つまり結果をシフトして，WORK 1 ≧ WORK 2なら最下位ビット1，そうでなければ0とする。そして，引くことができた場合はWORK 1をWORK 2だけ減らし，WORK 2は，1を立てたのだから，1加えてやる。しかし，これは上位バイトへの繰り上がりが出ないので，最下位バイトだけをインクリメントすればよい。また0なら，WORK 2の最下位ビットを0にする。これは，もともと1だったのだから，DECでよい。

この操作を8回繰り返せば整数部は求められる。しかし整数部だけではつまらないので小数部も求めることにする。すると，平方根を求めたい数は整数ということにしているので，WORK 1には0を送ればよいが，このときすでにXレジスタで示されるアドレスからの2バイトは0なので，そのまま続ければよい。

さて，これは整数部1バイト，小数部1バイトの例だが，もっと桁数を多く求めたい人も多いと思う。例えば，小数部に256バイト使えば10進数換算で700桁以上となる。このときの変更は，WORK 1とWORK 2を求めたい桁数に応じて大きくすればよい。じっくり考えれば，これらは求めたい小数点以下のバイト数＋2程度でよいことがわかる。**リスト6**ではループ回数をBレジスタとしているが，これは求めるビット数なので，結果を32バイト以上にすると，Bレジスタでは間に合わないので，メモリを使うようにすべきだ。また，ワークエリアも，この程度ならスタックでよかったが，大きくなると，そのエリアを固定すべきだ。そしてシフトなどはループを使う（**リスト6**では4回だったのでループはやめた）。

このようにすれば好きなだけ求められるが，10進数にしなくては意味がない。これはどうすればよいのか。それは簡単で，小数部を10倍して，整数部に出てきた分を表示していけばよい。10倍は2倍＋8倍だから，1回シフトしたものと3回シフトしたものを加えればよい。

他の関数について

今まで整数型（または固定小数点）で話を進めてきた。しかし，多くの関数はこれでは不都合である。そこで，浮動小数点を使う必要がある。浮動小数点まで話を進めると長くなるのでここでは触れないが，関数の計算はBASICの内部ルーチンを使えばいいだろう。もし，倍精度の関数がほしい場合は，テイラー展開を用いて乗・除算のみにして，内部ルーチンを使う方がよい。

また，初等関数でも三角関数程度は3Dパッケージに欠かせず，関数の値も範囲が決まっているので，固定小数点を用いる場合も多い。三角関数は周期関数なので，これは計算するより表をメモリに持つ方がよい。この辺については先号で田中氏が書いているので，そちらを参照するとよいだろう。

最　後　に

今までに挙げたプログラムはリロケータブルに作っているので，メモリのどこに置いても構わない。個人的にいろいろ活用してほしいと思う。できるだけ高速化したいときはワークエリアをダイレクトページなどに置くのもよいが，他との兼ね合いを考えなければならない。

※　　※　　※

今回は，紙面の都合で演算のプログラム例しか挙げられなかったが，次の機会にインデックスの実例としてソートプログラムを取りあげてみたい。また画面の操作についても，他の筆者が今号で解説しているが，私なりの方法を解説してみようと思っている。

マシン語特集

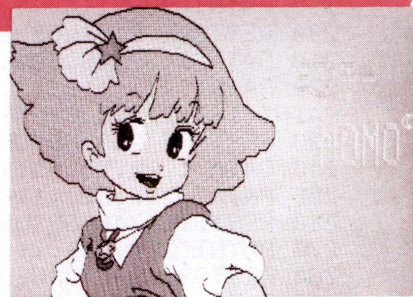
マシン語プログラム実例集 2

中間色PAINTもどきプログラム

川村　誠一

Ⓒ 葦プロダクション，ミンキーモモ

ＦＭシリーズの元祖，ＦＭ-８が出たとき，「これはスゴイ！どんなグラフィックをやろうか!?」などとまるで初恋の女の子にでも出会ったときの様に胸をワクワクさせたものでした。画面のキメ細かさといい，ペイントの速さといい申し分なく，つまらない絵を描いては自己満足にひたっていました。しかし，そんな僕をあざ笑うがごとく世の中は着実に進歩し，解像度，機能を増したマイコンが次々に産声をあげました。しかし，某社の09マイコン時代から09にのめりこんでしまった僕としては，そう簡単には09とは手が切れそうにもありません。ならばいっそのことＦＭの機能を増やしてやろうと考えたわけです。

今回の中間色ペイントルーチンは，はっきり言って不完全ではありますが，いまのところは僕の欲求を満たしてくれているので，Ｖ.１.０ということで発表させてもらいたいと思います。

アルゴリズム

リスト１を見てください。中間色のデータをマシン語のプログラムに送るために，USR関数を利用しています。今回のプログラムではデータを文字変数として引き渡しているので，USR関数でマシン語をコールしたとき，Aレジスタには３（文字変数なので），Xレジスタに文字変数のおかれているトップアドレスが入ります。さらにXレジスタの示しているアドレスから，はじめに文字変数の個数，次からは文字変数（１バイト）という順番でメモリ上に並んでいます。ですから，マシン語のプログラムでそれらデータを自分の好きなように処理できます。今回は単純に中間色のメインプログラムのうしろに転送しているだけです。**リスト１**では，A$＝USR0(TILE$)としてコールしていますが，このときAレジスタには３，XレジスタにはTILE$のおかれているトップアドレスが入るわけです。そしてマシン語のプログラムを実行しているのです。データは４バイトを１ブロックとして，フラグ，レッド，グリーン，ブルーの各情報に分かれます。フラグは０以外でデータを示し，０でデータエンドとなります。レッド，グリーン，ブルーは8 bitのビットパターンを16進２桁で示したものです。**図１**を見てください。図の様なビットパターンの組み合わせで「赤白赤白…」というドットパターンになります。ですから，1，FF，55，55，というデータでピンク色が出せるわけです。実際には，「赤白赤白…」のデータと「白赤白赤…」のデータの２種類のデータとで交互にペイントした方がよりピンク色に見えますが，そこは各人のお好みにデータを作ってみてください。

図１

次にメインのマシン語のプログラムの説明をしましょう。全体的には**図２**の様なフローで処理します。まずUSR関数によりデータの内容をXレジスタを使ってメインプログラムにリンクしてから，BIOSをコールしています。ここではサブシステムのメインテナンスコマンドを利用して，メインCPUとサブCPUの共通RAM領域上のプログラムを実行するようにしています。メインCPU上での共通RAMアドレスは$FC80～$FCFFの128バイトで，それはサブCPU上では$D380～$D3FFになります。ですから共通RAM上でかつサブCPU上で何か実行させるとき，充分にアドレスに注意してください。また，当然128バイト以内でなければいけません。**リスト２**のTABLEからENDまでが共通R

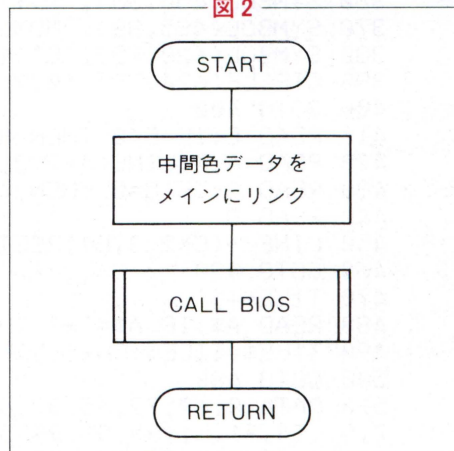
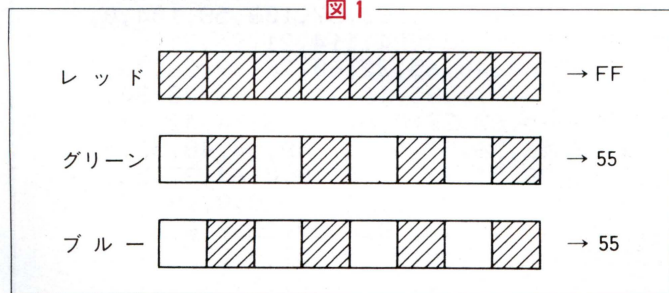



図２

リスト1　BASICプログラムリスト(for FM-7,8)

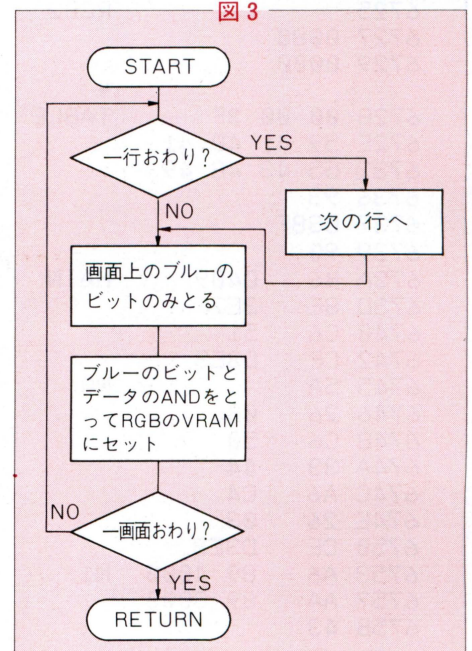

```
100 CLEAR300,&H66FF:DEF USR0=&H6700:COLOR7,0:CLS
110 LINE(0,0)-(639,199),PSET,7,BF
120 RESTORE 510:GOSUB 410
130 PAINT(0,0),3,0:PAINT(102,9),1,0:PAINT(120,26),1,0:PAINT(240,22),1,0:PAINT(15
9,160),1,0
140 RESTORE 150:GOSUB 470
150 DATA 1,FF,FF,88,1,FF,FF,22,*
160 PAINT(221,3),1,0:PAINT(164,158),1,0:PAINT(153,159),1,0:PAINT(170,163),1,0:PA
INT(144,163),1,0
170 RESTORE 180:GOSUB 470
180 DATA 1,FF,22,11,1,FF,44,77,*
190 PAINT(208,114),1,0
200 RESTORE 210:GOSUB 470
210 DATA 1,22,00,00,1,88,00,00,*
220 PAINT(222,118),2,0:PAINT(214,196),2,0:PAINT(176,150),2,0:PAINT(145,170),2,0:
PAINT(167,21),1,0:PAINT(284,131),1,0,6
230 RESTORE 240:GOSUB 470
240 DATA 1,00,55,55,*
250 PAINT(236,52),1,0:PAINT(301,63),1,0:PAINT(376,179),1,0:PAINT(168,120),1,0
260 RESTORE 740:GOSUB 410
270 PAINT(208,81),7: PAINT(280,87),7
280 RESTORE 290: GOSUB 470
290 DATA 1,FF,55,55,1,FF,AA,AA,*
300 RESTORE 700:GOSUB 410
310 PAINT(184,82),1,0:PAINT(270,87),1,0
320 RESTORE 330:GOSUB 470
330 DATA 1,00,22,00,1,00,88,00,*
340 LINE(194,92)-(195,92),PSET,7:LINE(271,96)-(271,97),PSET,7
350 PAINT(186,85),0: PAINT(267,90),0
360 SYMBOL(450,36),"ﾐﾝｷｰ",4,4,6
370 SYMBOL(485,80),"MOMO",4,4,5
380 SYMBOL(620,75),"C",1,1,7
390 CIRCLE(624,78),10,7,.5
400 GOTO 400
410 READ E:IF E<0 THEN RETURN
420 READ A,B:LINE(A*2.3,B)-(A*2.3,B),PSET,E
430 READ C:IF C=0 THEN 410
440 READ D
450 LINE -(C*2.3,D),PSET,E
460 GOTO 430
470 TILE$=""
480 READ A$:IF A$="*" THEN A$=USR0(TILE$):RETURN
490 TILE$=TILE$+CHR$(VAL("&H"+A$))
500 GOTO 480
510 DATA 0,69,72,65,82,65,85,65,88,65,93,66,95,67,101,64,100,63,99,60,95,59,93,5
7,91,54,91,51,93,51,99,52,101,53,103,57,109,60,110,61,111,65,111,66,110,68,114,7
1,117,75,120,78,122,80,123,83,125,95,129,96,129,101,129,105,128,113,125,115,124
520 DATA 120,122,125,114,125,104,126,105,130,100,129,97,0,0,126,99,127,99,0,0,12
6,99,137,88,0,0,126,99,137,81,0,0,147,55,149,60,149,73,144,82,142,82,142,78,140,
80,136,80,136,76,131,80,129,78,132,66,132,56,130,48,0,0,132,56,128,72,124,78
530 DATA 123,78,123,75,120,78,117,78,117,72,120,68,0,0,117,72,116,72,116,68,0,0,
119,63,111,75,108,75,108,73,102,76,98,76,98,72,102,62,105,45,99,59,89,72,87,72,8
7,70,0,0,88,68,84,71,81,71,81,67,71,74,71,72,75,67,69,71,66,73,0,0,106,107,107
540 DATA 108,107,110,106,110,0,0,88,112,90,113,101,113,107,115,0,0,89,114,92,118
,99,121,102,121,103,120,104,119,105,116,106,115,0,0,96,124,97,125,100,125,101
550 DATA 124,0,0,92,117,98,117,101,120,101,121,0,0,92,114,98,115,102,116,106,115
,0,0,55,96,57,96,58,97,0,0,62,103,62,101,59,99,58,99,0,0,57,100,57,103,58,104,0,
0,78,33,90,31,102,31,117,33,139,41,142,43,140,38,130,28,124,24,114,21,96,21,90
560 DATA 22,80,26,0,0,76,26,83,28,77,31,79,40,72,35,64,40,65,32,60,28,69,26,73,1
8,76,26,0,0,70,23,60,14,49,8,40,8,38,10,38,14,39,15,37,15,35,16,31,19,30,20,30,2
3,34,27,32,28,32,30,35,31,0,0,34,34,38,29,49,25,59,25,63,27,0,0,74,35,74,42,77
570 DATA 53,74,53,71,51,68,57,64,59,60,59,58,58,57,56,56,57,52,57,48,54,48,51,42
,51,37,46,37,41,35,41,35,35,0,0,69,25,55,18,0,0,62,28,54,32,52,35,0,0,64,37,58,4
7,0,0,82,62,83,60,89,59,92,62,0,0,126,67,128,65,130,65,133,67,133,72,0,0,181,78
580 DATA 173,75,163,67,146,32,134,16,118,6,106,3,104,2,90,2,70,6,59,12,0,0,32,25
,20,27,17,27,4,25,3,25,3,28,6,32,3,32,3,34,5,36,1,38,1,47,8,57,10,57,0,0,7,57,7,
61,13,67,18,67,11,73,11,82,16,90,22,93,22,99,25,103,32,105,32,112,42,119,49,119
```

```
590 DATA 53,125,61,127,0,0,181,78,183,81,183,89,177,97,173,97,173,99,170,101,164
,101,164,103,161,105,159,104,162,109,163,113,163,117,155,123,145,123,141,131,136
,133,0,0,72,117,74,124,74,126,73,126,68,127,68,128,73,130,90,129,93,128,0,0,74
600 DATA 126,72,130,0,0,116,124,121,125,122,126,121,130,0,0,74,124,63,123,61,126
,61,130,67,140,0,0,63,133,63,137,0,0,78,150,90,148,104,140,0,0,98,149,104,140,11
1,134,120,130,126,130,134,132,140,138,140,143,134,140,129,140,122,143,115,149,0
610 DATA 0,64,141,60,142,56,144,52,148,51,150,0,0,103,142,94,149,0,0,63,137,65,1
47,68,150,74,150,0,0,140,143,145,148,0,0,137,144,141,147,150,150,0,0,116,150,114
,154,112,156,111,160,110,164,108,170,108,176,109,178,110,180,114,180,0,0,110
620 DATA 176,114,180,120,186,0,0,114,174,120,186,124,199,125,200,0,0,99,176,100,
180,104,183,110,186,118,198,0,0,150,150,156,152,164,154,167,156,167,158,167,158,
166,160,168,161,170,162,174,166,176,170,176,178,174,182,172,184,0,0,150,176,158
630 DATA 176,164,178,168,180,170,182,172,185,174,188,178,196,179,200,0,0,153,175
,154,177,0,0,154,174,155,177,0,0,154,173,157,177,0,0,157,174,158,177,0,0,160,175
,160,177,0,0,164,178,158,179,156,180,152,182,150,184,148,190,147,195,148,198
640 DATA 149,200,0,0,138,183,136,184,135,186,136,190,140,193,145,195,147,195,0,0
,51,150,51,154,50,156,48,160,48,162,49,163,0,0,49,162,49,163,40,170,34,174,33,17
5,34,180,34,182,30,188,30,196,32,200,0,0,56,156,57,156,58,157,55,165,54,167,52
650 DATA 178,51,180,50,182,50,184,52,190,53,191,54,193,56,196,58,199,59,200,64,2
00,76,197,90,196,90,195,94,195,99,197,108,197,118,200,121,200,124,199,0,0,50,156
,51,159,52,162,54,167,0,0,56,160,58,160,0,0,55,161,58,162,0,0,54,162,58,164,0
660 DATA 0,46,170,44,172,44,176,45,178,47,180,50,182,0,0,40,183,42,185,45,186,47
,186,50,184,0,0,58,199,55,200,0,0,88,192,86,190,90,190,95,191,107,196,111,198,0,
0,148,198,144,199,141,199,0,0,142,193,142,194,0,0,144,192,144,194,0,0,145,192
670 DATA 146,194,0,0,94,150,90,156,87,160,82,168,78,176,74,170,72,166,0,0,100,14
5,90,150,80,155,74,158,0,0,69,150,68,152,68,158,0,0,65,150,64,153,66,158,0,0,68,
158,66,158,64,160,63,161,62,162,61,164,60,166,62,169,66,170,70,168,72,166,74
680 DATA 164,75,162,74,160,74,158,71,157,68,158,0,0,68,158,74,150,76,149,77,149,
78,150,77,152,72.5,158,0,0,62,169,61,172,62,175,64,176,66,177,70,176,73,173,73,1
72,71,171,70,172,67,173,66,173,65,172,65,171,68,170,0,0,63,161,65,164,63,167,67
690 DATA 166,70,168,70,164,74,162,70,161,70,158,66,161,63,161,0,-1
700 DATA 0,64,73,64,66,0,0,74,76,76,79,79,80,0,0,72,79,73,80,77,81,0,0,72,83,77,
83,0,0,73,86,75,85,0,1,78,82,76,86,76,89,76,92,77,95,80,97,82,97,84,96,87,88,86,
82,84,80,0,0,78,79,83,76,89,74,0,0,88,76,91,76,95,77,0,0,85,77,91,77,0,0,84,78
710 DATA 93,78,0,0,79,80,84,79,0,0,90,79,94,79,0,0,80,80,81,80,0,0,86,99,91,99,0
,1,114,87,112,89,110,97,111,100,113,101,115,102,116,102,117,101,121,94,121,87,11
9,85,0,0,126,86,128,86,0,0,114,86,118,85,0,0,125,85,128,85,0,0,116,84,127,84,0
720 DATA 0,119,83,127,83,0,0,116,83,119,81,126,81,129,82,0,0,128,85,133,82,0,0,1
28,86,132,86,133,85,0,0,126,86,129,88,0,0,127,89,129,91,0,0,118,103,123,103,0,0,
81,83,78,86,79,91,81,91,82,88,82,84,81,83,0,0,116,88,114,90,114,94,116,94,118
730 DATA 91,118,89,116,88,0,1,81,91,79,92,81,94,82,94,83,93,83,90,81,89,0,1,116,
94,114,96,115,99,116,99,117,96,116,94,0,-1
740 DATA 7,93,79,95,86,94,92,92,95,90,98,80,98,78,96,84,79,93,79,0,7,125,85,127,
94,124,100,122,102,116,102,112,96,120,85,125,85,0,-1
```

AM上にきます(ただし, ENDから20バイト程が中間色データとなります)。メインでは, まずVRAMのアクセスを可能とするため, サブCPUの$D409をリードします。次にXレジスタにブルーのVRAMのエンドアドレスをロードし, Bレジスタに1ライン分プラス1をロードします。またUレジスタには中間色データのトップアドレスをロードします。M2より実際に中間色のデータに変換していきます。

リスト2と**図3**のフローを見てください。結局, まず画面上のブルーのビットパターンのみに注目し, そのパターンと中間色のデータとANDをとってるだけなのです!? すなわち, 中間色を出したい所をまずブルーでペイントしてからこのルーチンをコールしなければならないのです。なぜブルーにしたかというと, VRAMのトップであることと, あまり使わない色だからです(ちなみにサブCPUでは, 若いアドレスからブルー, レッド, グリーンの順に16Kバイトずつ VRAMがあります)。これが「中間色ペイントもどき」の名前の由来なのです。ですから, いろいろな中間色を出す際, ブルー系統の中間色は終わりの方でしか使えないわけです。まあ, 今のところ不便は感じていないのですが, やはりどうも…。

ペイントのスピードは, まあまあです。ただ, ほんの少しの所をペイントする際も全画面分ペイントしていることになるのでもったいないと言えばもったいない気がしますが, 思ったとおりの中間色が出たときの気持ちはまた格別なものです。皆さんもいろいろな中間色を作って, 楽しんでみてください。

図3

使用方法

まず**リスト2**のマシン語の部分（一番左がアドレスでその次の1～4バイトがマシン語）を入力して，セーブしてください。リストでは$6700からになっていますが，ポジションインデペンデントに作ってありますので好きなアドレス，空いているアドレスにおいて結構です（セーブの際は，SAVEM "ファイルネーム", &H6700, &H678D, &H6700として，ロード時にオフセットをつけてリロケートしてください。）

このルーチンをコールするときは，**リスト1**の様にまずプログラムの冒頭でDEF USR0＝&H××××（××××は**リスト2**のSTARTアドレス）と宣言して，（もちろんCLEAR, &H××××－1も忘れずに!!）コールする前に塗りたい部分をブルーでペイントしておいてください。中間色のデータは必ず4バイト必要ですから間違いのない様に!!　また，共通RAM上は128バイトしかないので，中間色データは最大24個まででですのでこれも注意してください。

中間色のデータは**図1**の様なものを用意して作ると便利ですが，やはり実際に画面に出してみないと，こうと思った色はなかなか出ません。カットアンドトライの精神で頑張ってください。

おわりに

近いうちに完全な形の中間色ペイントプログラムを（BASICのコマンドの一つとして!）完成させようと思っていますのでご期待のほどを!　モモちゃんのデータを作ってくれた，元樹君，つかさちゃんにこの場を借りてお礼を申し上げます。

リスト2

```
                        *
                        *  MIXED COLOR PAINT ROUTINE     コメント
                        *
6700                           ORG    $6700              $6700からプログラム作成
            FBFA        BIOS   EQU    $FBFA              BIOSというラベルは$FBFAである
                        *
            6700        START  EQU    *                  STARTというラベルは現在のプログラムカウンタである
6700 34     56          PAINT  PSHS   D,X,U              このルーチンで使用するレジスタの退避
6702 E6     84                 LDB    ,X                 Bレジスタに文字変数の個数をロード(中間色データの個数)
6704 EE     01                 LDU    1,X                UレジスタにX+1のアドレスをロード
6706 30     8D 0084            LEAX   END,PCR            XレジスタにENDの相対アドレスをロード
670A A6     C0          P0     LDA    ,U+                AレジスタにUレジスタのアドレスの内容をロード後U=U+1
670C A7     80                 STA    ,X+                XレジスタのアドレスにAレジスタの内容をストア後X=X+1
670E 5A                        DECB                      B=B－1
670F 26     F9                 BNE    P0                 IF B≠0　THEN P0
6711 4F                        CLRA                      A=0
6712 A7     84                 STA    ,X                 XレジスタのアドレスにAレジスタの内容をストア
6714 30     8C 14              LEAX   <TABLE,PCR         XレジスタにTABLEの相対アドレスをロード
6717 AF     8C 0B              STX    <RCB2,PCR          Xレジスタの内容をRCB2にストア
671A 30     8C 06              LEAX   <RCB1,PCR          XレジスタにRCB1の相対アドレスをロード
671D AD     9F FBFA            JSR    [BIOS]             BIOSを間接コール
6721 35     D6                 PULS   PC,D,X,U           レジスタの復帰とBASICへのリターン
6723 10                 RCB1   FCB    16                 BIOSサブアウトコマンドコード
6724 00                        FCB    0                  1バイト定数セット
6725                    RCB2   RMB    2                  転送するプログラムのトップアドレス(TABLE)
6727 0080                      FDB    128                転送するプログラムのバイト数
6729 0000                      FDB    0                  2バイト定数セット
                        *
672B 00 00 3F           TABLE  FCB    $00,$00,$3F        メインテナンスコマンドコード
672E 59 41 4D 41               FCC    /YAMAUCHI/         キーワード
6732 55 43 48 49
6736 93                        FCB    $93                メインテナンスコマンド中のマシン語サブルーチンコール
6737 D38F                      FDB    MAIN-TABLE+$D380   MAINのサブCPU上での絶対アドレス
6739 90                        FCB    $90                メインテナンスコマンド中のリターン
673A B6     D409        MAIN   LDA    $D409              サブシステムのVRAMアクセスON
673D 8E     3E7F               LDX    #$3E7F             ブルーVRAMの最後のアドレス
6740 C6     51                 LDB    #81                1ライン分のバイト数+1，B=81
6742 CE     D3E3               LDU    #END-TABLE+$D380   中間色データのサブCPU上での絶対アドレスをUレジスタにロード
6745 5A                 M0     DECB                      B=B－1
6746 26     0B                 BNE    M1                 IF B≠0　THEN M1
6748 C6     50                 LDB    #80                B=80
674A 33     44                 LEAU   4,U                U=U+4
674C A6     C4                 LDA    ,U                 AレジスタにUレジスタのアドレスの内容をロード
674E 26     03                 BNE    M1                 IF A≠0　THEN M1
6750 CE     D3E3               LDU    #END-TABLE+$D380   中間色データのサブCPU上での絶対アドレスをUレジスタにロード
6753 A6     89 4000     M1     LDA    $4000,X            AレジスタにレッドVRAMの内容をロード
6757 AA     89 8000            ORA    $8000,X            AレジスタとグリーンVRAMの内容のOR(論理和)をとる
675B 43                        COMA                      Aレジスタの各ビットを反転
```

```
675C A4    84                 ANDA   ,X        Aレジスタとブルー VRAMの内容のAND(論理積)をとる
675E 27    26                 BEQ    M2        IF A=0  THEN M2
6760 34    02                 PSHS   A         Aレジスタをシステムスタックに退避
6762 A4    41                 ANDA   1,U       AレジスタとUレジスタ+1のアドレスの内容(中間色のレッド情報)とANDをとる
6764 AA    89 4000            ORA    $4000,X   AレジスタとレッドVRAMの内容とORをとる
6768 A7    89 4000            STA    $4000,X   Aレジスタの内容をレッドVRAMにストアする
676C A6    E4                 LDA    ,S        スタック中の元Aレジスタの内容をAレジスタにロード
676E A4    42                 ANDA   2,U       AレジスタとUレジスタ+2のアドレスの内容(中間色のグリーン情報)とANDをとる
6770 AA    89 8000            ORA    $8000,X   AレジスタとグリーンVRAMの内容とORをとる
6774 A7    89 8000            STA    $8000,X   Aレジスタの内容をグリーンVRAMにストアする
6778 A6    84                 LDA    ,X        AレジスタにブルーVRAMの内容をロード
677A A0    E4                 SUBA   ,S        Aレジスタからスタック中の元のAレジスタの内容を引く
677C A7    84                 STA    ,X        Aレジスタの内容をブルーVRAMにストアする
677E A6    E0                 LDA    ,S+       スタック中の元のAレジスタの内容をAレジスタにロード後S=S+1(=PULS A)
6780 A4    43                 ANDA   3,U       AレジスタとUレジスタ+3のアドレスの内容(中間色のブルーの情報)とANDをとる
6782 AA    84                 ORA    ,X        AレジスタとブルーVRAMの内容のORをとる
6784 A7    84                 STA    ,X        Aレジスタの内容をブルーVRAMにストアする
6786 30    1F        M2       LEAX   -1,X      X=X-1
6788 8C    FFFF               CMPX   #$FFFF    Xレジスタの内容と$FFFFとの比較
678B 26    B8                 BNE    M0        IF X≠$FFFF  THEN M0
678D 39                       RTS              リターン
                     *
678E                 END      RMB    128-END+TABLE 中間色データ領域
                     *
                              END    PAINT     PAINTプログラムの終了

0 ERROR(S) DETECTED                            エラーが発生しなかったというリポート

SYMBOL TABLE:                                  ラベル(シンボル)名とそのアドレス表

BIOS   FBFA   END    678E   M0     6745   M1     6753   M2     6786
MAIN   673A   P0     670A   PAINT  6700   RCB1   6723   RCB2   6725
START  6700   TABLE  672B
```

マシン語特集

マシン語プログラム実例集 3

サブシステムプログラムの転送・実行

石原　宏明

FM-8で画面を直接制御しようとするとき，VRAMがメインCPUの管理下にないため，LEVEL-3などよりもかなり面倒な手続きが必要です。マシン語の勉強には画面に何か動くものを表示させた方が興味もわき，いちばん良い方法です。

そこで高速ゲームを作ったり，画面制御のアルゴリズムを勉強するときのテストラン用にこのプログラムを作りました。後にモンスター2匹が2：3の速度比で移動していくデモプログラムを付けました。画面制御のアルゴリズムの勉強の参考になれば幸いです。

使い方

まずサブシステムに転送して実行すべきプログラム（以下，画面制御プログラムと呼びます）を作っておき，それを$7000を開始番地として格納しておきます（このプログラムをRUNさせてからでもLOADできます）。その際，CLEAR，&H6FFFを事前に実行しておいた方がいいでしょう。また，画面制御プログラムのバイト数を確認しておいてください。

次にサブシステムプログラムを入力し，RUNさせます。するとマシン語ルーチンを書き込んだあと画面制御プログラムがちゃんと$7000以降にあるかどうか聞いてくるので，y（またはY）かn（またはN）を入力してください。n（またはN）を入力した場合，画面制御プログラムを外部記憶装置から，LOADできます。

次に画面制御プログラムのバイト数を入力します。これは少なすぎては困りますが，4Kバイトまでなら多くても問題ありません。

次に画面をクリアするかどうか聞いてきます。yかY，nかNで答えてください。

最後にリターンキーを入力するとマシン語ルーチンに制御が移り，画面制御プログラムのサブ側への受け渡しと実行を行います。

プログラムについて

プログラムの主要部はマシン語でできており，BASICがマシン語ルーチンにわたす引数は画面制御プログラムのバイト数だけです。これも4Kバイトを越えなければ多少多くてもかまわないので，そのあたりを改良した方がよかったかもしれません。

マシン語ルーチンはポジションインディペンデントにできており，BASICのDATA文の形でプログラム中にあります。行番号20～60行でMAを先頭アドレスとして格納されています。20行のMA＝&H6000を書き換えれば，自由な位置に置くことができます。

サブへの転送と実行のアルゴリズム

BIOSとかサブCPUのHALT，RESTART，共通領域のアドレス計算，などという面倒なことを一切考えなくてもいいようにとこのプログラムを作ったのですが，動作原理の説明のためにプログラムを再び見てください。

原理は非常に簡単で，画面制御プログラムを100バイトずつ区切って共通領域経由でサブの$C000以降にどんどん転送し，転送が終了した時点で再びBIOSをコールしてサブの$C000以降のプログラムを実行するものです。たったこれだけのことです。ただプログラムは，ポジションインディペンデントにするため少しばかり工夫されていますが。

VRAMを直接アクセスするプログラムは，当然サブCPUの管理下にないといけません。そこでそのような画面制御プログラムをサブのRAM上に送って制御をそのプログラムに移せば，画面に絵が描けることになります。メインからVRAMに直接データが送れないのと同様，画面制御プログラムもサブのRAM上に直接ストアできません（直接というのはMONコマンドを使ってもメインのRAM上にはストアができるが，サブのRAM上にはストアできないということです）。

そこで共通領域を使って一度に少しずつこま切れで送ることになります。共通領域とは物理的には一つであるのにメイン側では$FC80～$FCFF，サブ側では$D380～$D3FFという二重のアドレスのついた128

サブシステムプログラム転送・実行(FM-7, 8用)


```
10 CLEAR,&H5FFF
20 MA=&H6000
30 FOR I=MA TO MA+237
40   READ A$:A=VAL("&H"+A$)
50   POKE I,A
60 NEXT
70 PRINT"SUB SYSTEM ﾍ ﾃﾝｿｳ ｽﾙ PROGRAM ﾊ"
80 PRINT"&H7000 ｲｺｳ ﾆ ｱﾘﾏｽｶ ? ";
90 A$=INPUT$(1):IF A$<>"y" AND A$<>"Y" AND A$<>"n" AND A$<>"N" THEN 90
100 PRINT A$
110 IF A$="y" OR A$="Y" THEN 150
120 PRINT"ｿﾚﾃﾞﾊ LOADM ｼﾃ ｸﾀﾞｻｲ"
130 LINEINPUT"FILE NAME ";FI$
140 LOADM FI$
150 INPUT"ﾅﾝ byte ﾃﾞｽｶ ";BYTE
160 POKE MA+84,BYTE ¥ 256
170 POKE MA+85,BYTE MOD 256
180 PRINT"ｶﾞﾒﾝ ﾊ ｸﾘｱ ｼﾏｽｶ ? ";
190 A$=INPUT$(1):IF A$<>"y" AND A$<>"Y" AND A$<>"n" AND A$<>"N" THEN 190
200 PRINT A$
210 PRINT"RETURN KEY ﾃﾞ SUB SYSTEM ﾃﾝｿｳ ﾄ ｼﾞｯｺｳ ｦ ｵｺﾅｲﾏｽ"
220 WHILE INKEY$<>CHR$(13):WEND
230 IF A$="y" OR A$="Y" THEN CLS
240 EXEC MA
250 DATA 30,8C,65,AF,8C,54,30,8D,00,D5,AF,8C,55,CC,C0,00
260 DATA ED,8C,63,CC,70,00,ED,8C,3D,AE,8C,3A,31,8C,5C,C6
270 DATA 64,A6,80,A7,A0,5A,26,F9,30,8C,2D,AD,FF,FB,FA,EC
280 DATA 8C,44,C3,00,64,ED,8C,3E,EC,8C,1B,C3,00,64,ED,8C
290 DATA 15,EC,8C,10,83,00,64,ED,8C,0A,2A,CD,30,8C,11,AD
300 DATA FF,FB,FA,39,00,00,00,00,10,00,00,00,00,77,00,00
310 DATA 10,00,00,00,00,0F,00,00,00,00,3F,59,41,4D,41,55
320 DATA 43,48,49,91,D3,93,00,00,00,64,90,00,00,00,00,00
330 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
340 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
350 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
360 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
370 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
380 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
390 DATA 00,3F,59,41,4D,41,55,43,48,49,93,C0,00,90
```

サブシステムプログラム転送・実行のデータ部分の解説

```
0001  6000                    ORG   $6000
0002  6000           BIOS    =$FBFA
0003  6000 308C65             LEAX <DATAM,PC    ) 転送用データ先頭アドレスセット
0004  6003 AF8C54             STX  <PMLM+2,PC   )
0005  6006 308D00D5           LEAX DATAE,PC     ) 実行用データ先頭アドレスセット
0006  600A AF8C55             STX  <PMLE+2,PC   )
0007  600D CCC000             LDD  #$C000       ) 転送先アドレスセット
0008  6010 ED8C63             STD  <MADR,PC     )
0009  6013 CC7000             LDD  #$7000       ) 画面制御プログラム開始番地セット
000A  6016 ED8C3D             STD  <PADR,PC     )
000B  6019 AE8C3A    LOOP1   :LDX  <PADR,PC     )
000C  601C 318C5C             LEAY <PAREA,PC    )
000D  601F C664               LDB  #$64         ) 画面制御プログラム 100 byte分を
000E  6021 A680      LOOP2   :LDA  ,X+          ) 転送用データ領域PAREAにセット
000F  6023 A7A0               STA  ,Y+          )
0010  6025 5A                 DECB              )
0011  6026 26F9               BNE  LOOP2        )
0012  6028 308C2D             LEAX <PMLM,PC     ) BIOSコール   100byteの転送
0013  602B ADFFFBFA           JSR  [BIOS]       )
0014  602F EC8C44             LDD  <MADR,PC     )
0015  6032 C30064             ADDD #$0064       ) 転送先アドレスを100ふやす
0016  6035 ED8C3E             STD  <MADR,PC     )
0017  6038 EC8C1B             LDD  <PADR,PC     )
0018  603B C30064             ADDD #$0064       ) 画面制御プログラムの次の100byte開始番地をセット
0019  603E ED8C15             STD  <PADR,PC     )
```

```
001A  6041 EC8C10          LDD   <BYTE,PC      カウンタを100減らす
001B  6044 830064          SUBD  #$0064
001C  6047 ED8C0A          STD   <BYTE,PC
001D  604A 2ACD            BPL   LOOP1         転送すべきデータがまだあればLOOP1へ
001E  604C 308C11          LEAX  <PMLE,PC      BIOSコール   $C000から実行
001F  604F ADFFFBFA        JSR   [BIOS]
0020  6053 39              RTS
0021  6054          BYTE  :RMB   2             画面制御のプログラムのバイト数
0022  6056          PADR  :RMB   2             画面制御プログラムの転送すべき100byte分の先頭アドレス
0023  6058 1000     PMLM  :FDB   $1000         サブシステム出力コマンド      (初期値 $7000)
0024  605A                 RMB   2             転送用データ先頭アドレス
0025  605C 0077            FDB   119           転送用データバイト数
0026  605E 0000            FDB   $0000
0027  6060 1000     PMLE  :FDB   $1000         サブシステム出力コマンド
0028  6062                 RMB   2             実行用データ先頭アドレス
0029  6064 000F            FDB   15            実行用データバイト数
002A  6066 0000            FDB   $0000
002B  6068 0000     DATAM :FDB   $0000         転送用データ
002C  606A 3F              FCB   $3F
002D  606B 59414D4155      FCC   /YAMAUCHI/
           434849
002E  6073 91              FCB   $91
002F  6074 D393  119 byte  FDB   $D393
0030  6076          MADR  :RMB   2             転送先アドレス(初期値 $C000)
0031  6078 0064            FDB   $0064
0032  607A 90              FCB   $90
0033  607B          PAREA :RMB   $64           画面制御プログラム100byte分格納領域
0034  60DF 0000     DATAE :FDB   $0000         実行用データ
0035  60E1 3F              FCB   $3F
0036  60E2 59414D4155      FCC   /YAMAUCHI/
           434849
0037  60EA 93    15byte    FCB   $93
0038  60EB C000            FDB   $C000
0039  60ED 90              FCB   $90
```

バイトのRAMのことです。たとえばメイン側で＄FC80に＄12を書き込めば，サブ側では＄D380に＄12が書き込まれたことになります。ここの128バイトがメイン，サブ間のデータ受け渡し場所になります。

共通領域にはサブが仕事中はデータを書き込んではなりません。サブが仕事を終えるのを待ってサブをいったん止め，データを書き込み再びサブを動かす，という面倒なことをしなくてはなりませんが，BIOSを使えばこのようなことはすべてBIOSが管理してやってくれます。そこでわれわれのする仕事は共通領域のアドレス計算と，BIOSの使用です。

画面制御プログラムの転送先は，＄C000～＄CFFFの4KバイトのRAMにします(単純な計算でわかることですが，B, R, Gの各VRAMの終りにも384バイトのフリーエリアがあります。ここに転送してもかまいませんが少し狭すぎます)。

BIOSにより次のようなフォーマットでデータをサブ側に送ると，サブは＄D393からの100バイトのデータを＄C000以降に転送します。

00003F YAMAUCHI 91D393C0000064
(コマンド　キーワード　サブコマンド　転送元アドレス　転送先アドレス　バイト数)

90データ～
(サブコマンド　100バイトの転送すべきデータ(画面制御プログラム100バイト分))

＄91：サブシステム内のブロック転送コマンド
＄91転送元アドレス(2バイト)転送先アドレス(2バイト)バイト数(2バイト)
のようにして使う。

＄90：終了コマンド

このあとさらに送るべきデータがあれば転送先アドレスを100だけプラスし，再び同じフォーマットでデータを送ればよいのです。

画面制御プログラムをすべて転送したら次は＄C000からの実行です。それには，BIOSを使って次のようなフォーマットでデータをサブ側に送ればいいのです。

00003F YAMAUCHI 93C00090

コマンド　キーワード　サブコマンド　サブコマンド

$93：サブシステム内のJSRコマンド

$93サブルーチン開始アドレス（絶対アドレス2バイトで指定）

のようにして使う。

1回目の転送におけるデータの流れは図1のようになります。

順序が逆になってしまいましたが，最後にBIOSを使ってデータをサブ側へ送る方法を説明しましょう。BIOSを使うには8バイトのRCBをセットし，XレジをRCBの先頭にセットしてBIOSをコールすればよいのです。

例として，サブへ

00003FYAMAUCHI93C00090

を送る場合のRCBは次のようになります。

1000 アドレス（2 byte） 000F0000（バイト数2 byte）

00	00	3F	Y	A	M	A	U	C	H	I	93	C0	00	90

実際にコーディングすると次のとおりです。

```
        LEAX  RCB, PC
        JSR   〔BIOS〕
        RTS
RCB：FDB   $1000
     FDB   DATA
     FDB   15
     FDB   $0000
DATA：FDB  $0000
     FCB   $3F
     FCC   ／YAMAUCHI／
     FCB   $93, $C0, $00, $90
```

デモプログラム（モンスター移動）FM-7,8用

```
0001  7000                     ORG   $7000
0002  7000 B6D409              LDA   $D409       VRAMアクセスフラグをON
0003  7003 308D0123   INIT    :LEAX  DOT0,PC   ┐
0004  7007 108E5A40            LDY   #$5A40    │
0005  700B C610                LDB   #16       │
0006  700D 1700F0     LP1     :LBSR  POKE      │ 赤のモンスター 初期状態（パターン0）表示
0007  7010 31A850              LEAY  80,Y      │
0008  7013 5A                  DECB            │
0009  7014 26F7                BNE   LP1       ┘
000A  7016 308D0110            LEAX  DOT0,PC   ┐
000B  701A 108E9F40            LDY   #$9F40    │
000C  701E C610                LDB   #16       │
000D  7020 1700DD     LP2     :LBSR  POKE      │ 緑のモンスター 初期状態（パターン0）表示
000E  7023 31A850              LEAY  80,Y      │
000F  7026 5A                  DECB            │
0010  7027 26F7                BNE   LP2       ┘
0011  7029 308D014D   START   :LEAX  DOT1,PC   ┐
0012  702D AF8D00F1            STX   DAPR,PC   │ パターン1開始アドレスを赤と緑の2つのポインタにセット
0013  7031 AF8D00EF            STX   DAPG,PC   ┘
0014  7035 CC5A40              LDD   #$5A40    ┐ 赤のモンスター表示領域の左上隅のVRAMのアドレス
0015  7038 ED8D00EA            STD   VRAMR,PC  ┘
0016  703C CC9F40              LDD   #$9F40    ┐ 緑のモンスター表示領域の左上隅のVRAMのアドレス
0017  703F ED8D00E5            STD   VRAMG,PC  ┘
0018  7043 8603                LDA   #3        ┐ 赤のモンスター移動比（3回に1回）
0019  7045 A78D0030            STA   CTR,PC    ┘
001A  7049 8602                LDA   #2        ┐ 緑のモンスター移動比（2回に1回）
001B  704B A78D003A            STA   CTG,PC    ┘
001C  704F 8E04B0              LDX   #1200     ┐ メインループカウンターを1200回にセット
001D  7052 AF8D00CA            STX   CT,PC     ┘
001E  7056 8D12       LOOP    :BSR   MONR      ┐ モンスター移動ルーチンをコール
001F  7058 8D20                BSR   MONG      ┘
0020  705A 1700B8              LBSR  DLY         遅延ルーチン
0021  705D AE8D00BF   ここまで LDX   CT,PC     ┐
0022  7061 301F       がメイン LEAX  -1,X      │ 終了判定
0023  7063 AF8D00B9   ルーチン STX   CT,PC     │
0024  7067 26ED                BNE   LOOP      ┘
0025  7069 39                  RTS
0026  706A 6A8D000B   MONR    :DEC   CTR,PC    ┐ カウンタを−1し，0でなければ何もせずリターン
0027  706E 2608                BNE   RT1       ┘
0028  7070 8D18       赤色モンスター BSR PUTR  ………80byteの書き込みルーチン
0029  7072 8603       移動ルーチン LDA #3      ………カウンタを再び3にセット
002A  7074 A78D0001            STA   CTR,PC
002B  7078 39         RT1     :RTS
002C  7079            CTR     :RMB   1
002D  707A 6A8D000B   MONG    :DEC   CTG,PC    ┐ カウンタを−1し，0でなければ何もせずリターン
002E  707E 2608                BNE   RT2       ┘
002F  7080 8D43       緑色モンスター BSR PUTG  ………80byteの書き込みルーチン
0030  7082 8602       移動ルーチン LDA #2      ………カウンタを再び2にセット
0031  7084 A78D0001            STA   CTG,PC
0032  7088 39         RT2     :RTS
0033  7089            CTG     :RMB   1
```

```
0034   708A AE8D0094    PUTR   :LDX   DAPR,PC
0035   708E 10AE8D0093         LDY    VRAMR,PC
0036   7093 C610               LDB    #16
0037   7095 8D69        LP3    :BSR   POKE
0038   7097 31A850             LEAY   80,Y
0039   709A 5A                 DECB
003A   709B 26F8               BNE    LP3
003B   709D AE8D0081           LDX    DAPR,PC
003C   70A1 308850             LEAX   80,X
003D   70A4 AF8D007A           STX    DAPR,PC
003E   70A8 308D034E           LEAX   DEND,PC
003F   70AC AC8D0072           CMPX   DAPR,PC
0040   70B0 2612               BNE    JP1
0041   70B2 3089FD80           LEAX   -640,X
0042   70B6 AF8D0068           STX    DAPR,PC
0043   70BA AE8D0068           LDX    VRAMR,PC
0044   70BE 3001               LEAX   1,X
0045   70C0 AF8D0062           STX    VRAMR,PC
0046   70C4 39          JP1    :RTS
0047   70C5 AE8D005B    PUTG   :LDX   DAPG,PC
0048   70C9 10AE8D005A         LDY    VRAMG,PC
0049   70CE C610               LDB    #16
004A   70D0 8D2E        LP4    :BSR   POKE
004B   70D2 31A850             LEAY   80,Y
004C   70D5 5A                 DECB
004D   70D6 26F8               BNE    LP4
004E   70D8 AE8D0048           LDX    DAPG,PC
004F   70DC 308850             LEAX   80,X
0050   70DF AF8D0041           STX    DAPG,PC
0051   70E3 308D0313           LEAX   DEND,PC
0052   70E7 AC8D0039           CMPX   DAPG,PC
0053   70EB 2612               BNE    JP2
0054   70ED 3089FD80           LEAX   -640,X
0055   70F1 AF8D002F           STX    DAPG,PC
0056   70F5 AE8D002F           LDX    VRAMG,PC
0057   70F9 3001               LEAX   1,X
0058   70FB AF8D0029           STX    VRAMG,PC
0059   70FF 39          JP2    :RTS
005A   7100 A680        POKE   :LDA   ,X+
005B   7102 A7A4               STA    ,Y
005C   7104 A680               LDA    ,X+
005D   7106 A721               STA    1,Y
005E   7108 A680               LDA    ,X+
005F   710A A722               STA    2,Y
0060   710C A680               LDA    ,X+
0061   710E A723               STA    3,Y
0062   7110 A680               LDA    ,X+
0063   7112 A724               STA    4,Y
0064   7114 39                 RTS
0065   7115 8601        DLY    :LDA   #1
0066   7117 C601        LP5    :LDB   #1
0067   7119 5A          LP6    :DECB
0068   711A 26FD               BNE    LP6
0069   711C 4A                 DECA
006A   711D 26F8               BNE    LP5
006B   711F 39                 RTS
006C   7120             CT     :RMB   2
006D   7122             DAPR   :RMB   2
006E   7124             DAPG   :RMB   2
006F   7126             VRAMR  :RMB   2
0070   7128             VRAMG  :RMB   2
0071   712A             DOT0   :RMB   80
0072   717A             DOT1   :RMB   80
0073   71CA             DOT2   :RMB   80
0074   721A             DOT3   :RMB   80
0075   726A             DOT4   :RMB   80
0076   72BA             DOT5   :RMB   80
0077   730A             DOT6   :RMB   80
0078   735A             DOT7   :RMB   80
0079   73AA             DOT8   :RMB   80
007A   73FA             DEND   :EQU   *
```

(0034〜0046) 赤色モンスターを実際に表示するルーチン

- 0034: ドットパターン開始アドレスをセット
- 0035: 赤のモンスター表示領域左上隅のVRAMのアドレスをセット
- 0036: カウンタセット
- 0037: 横5 byteの書き込み
- 003B〜003D: ドットパターン開始アドレスを80ふやす
- 003E〜0040: パターン8を表示し終わったかどうか判定
- 0041〜0045: パターン8まで表示したので，ドットパターン開始アドレスをパターン1に戻し，VRAMの左上隅アドレスも1ふやす

(0047〜0059) 緑色モンスターを実際に表示するルーチン

- 005A: ドットパターンから連続5 byteを取り出し，VRAM上に転送する，横5 byteの書き込みルーチン
- 0065〜0066: 遅延ルーチン　この値をふやすと，モンスターの移動が遅くなる。現在最も速く，0の時が最も遅い。
- 006C: メインカウンタ
- 0071〜0079: パターン0〜パターン8の格納場所

デモプログラムダンプリスト

Add	+0	+1	+2	+3	+4	+5	+6	+7	Sum
7000	B6	D4	09	30	8D	01	23	10	:84
7008	8E	5A	40	C6	10	17	00	F0	:05
7010	31	A8	50	5A	26	F7	30	8D	:5D
7018	01	10	10	8E	9F	40	C6	10	:64
7020	17	00	DD	31	A8	50	5A	26	:9D
7028	F7	30	8D	01	4D	AF	8D	00	:3E
7030	F1	AF	8D	00	EF	CC	5A	40	:82
7038	ED	8D	00	EA	CC	9F	40	ED	:FC
7040	8D	00	E5	86	03	A7	8D	00	:2F
7048	30	86	02	A7	8D	00	3A	8E	:B4
7050	04	B0	AF	8D	00	CA	8D	12	:59
7058	8D	20	17	00	B8	AE	8D	00	:B7
7060	BF	30	1F	AF	8D	00	B9	26	:29
7068	ED	39	6A	8D	00	0B	26	08	:56
7070	8D	18	86	03	A7	8D	00	01	:63
7078	39	0B	6A	8D	00	0B	26	08	:74
7080	8D	43	86	02	A7	8D	00	01	:8D
7088	39	94	AE	8D	00	94	10	AE	:5A
7090	8D	00	93	C6	10	8D	69	31	:1D
7098	A8	50	5A	26	F8	AE	8D	00	:AB
70A0	81	30	88	50	AF	8D	00	7A	:3F
70A8	30	8D	03	4E	AC	8D	00	72	:B9
70B0	26	12	30	89	FD	80	AF	8D	:AA
70B8	00	68	AE	8D	00	68	30	01	:3C
70C0	AF	8D	00	62	39	AE	8D	00	:12
70C8	5B	10	AE	8D	00	5A	C6	10	:D6
70D0	8D	2E	31	A8	50	5A	26	F8	:5C
70D8	AE	8D	00	48	30	88	50	AF	:3A
70E0	8D	00	41	30	8D	03	13	AC	:4D
70E8	8D	00	39	26	12	30	89	FD	:B4
70F0	80	AF	8D	00	2F	AE	8D	00	:26
70F8	2F	30	01	AF	8D	00	29	39	:FE
7100	A6	80	A7	A4	A6	80	A7	21	:5F
7108	A6	80	A7	22	A6	80	A7	23	:DF
7110	A6	80	A7	24	39	86	01	C6	:77
7118	01	5A	26	FD	4A	26	F8	39	:1F
7120	00	00	00	00	00	00	00	00	:00
7128	00	00	00	00	00	00	00	00	:00
7130	00	00	00	00	00	0F	E0	00	:EF
7138	00	00	FF	FE	00	00	07	1F	:23
7140	F1	C0	00	1E	EF	EE	F0	00	:9C
7148	3F	BF	FB	F8	00	7F	1F	F1	:80
7150	FC	00	FE	0F	E0	FE	00	FE	:E5
7158	0F	E0	FE	00	FE	4F	E4	FE	:1C
7160	00	FF	1F	F1	FE	00	FF	FF	:0B
7168	FF	FE	00	1C	03	80	70	00	:0C
7170	08	01	00	20	00	00	00	00	:29
7178	00	00	00	00	00	00	00	00	:00
7180	00	00	00	00	00	07	F0	00	:F7
7188	00	00	7F	FF	00	00	03	8F	:10
7190	F8	E0	00	0F	77	F7	78	00	:CD
7198	1F	DF	FD	FC	00	3F	8F	F8	:BD
71A0	FE	00	7F	07	F0	7F	00	7F	:72
71A8	07	F0	7F	00	7F	27	F2	7F	:8D
71B0	00	7F	8F	F8	FF	00	7F	FF	:83
71B8	FF	FF	00	0E	01	C0	38	00	:05
71C0	04	00	80	10	00	00	00	00	:94
71C8	00	00	00	00	00	00	00	00	:00
71D0	00	00	00	00	00	03	F8	00	:FB
71D8	00	00	3F	FF	80	00	01	C7	:86
71E0	FC	70	00	07	BB	FB	BC	00	:E5
71E8	0F	EF	FE	FE	00	1F	C7	FC	:DC
71F0	7F	00	3F	83	F8	3F	80	3F	:37
71F8	83	F8	3F	80	3F	93	F9	3F	:44
7200	80	3F	C7	FC	7F	80	3F	FF	:BF
7208	FF	FF	80	07	00	E0	1C	00	:81
7210	02	00	40	08	00	00	00	00	:4A
7218	00	00	00	00	00	00	00	00	:00
7220	00	00	00	00	00	01	FC	00	:FD
7228	00	00	1F	FF	C0	00	00	E3	:C1
7230	FE	38	00	03	DD	FD	DE	00	:F1
7238	07	F7	FF	7F	00	0F	E3	FE	:6C
7240	3F	80	1F	C1	FC	1F	C0	1F	:99
7248	C1	FC	1F	C0	1F	C9	FC	9F	:1F
7250	C0	1F	E3	FE	3F	C0	1F	FF	:DD
7258	FF	FF	C0	03	80	70	0E	00	:BF
7260	01	00	20	04	00	00	00	00	:25
7268	00	00	00	00	00	00	00	00	:00
7270	00	00	00	00	00	00	FE	00	:FE
7278	00	00	0F	FF	E0	00	00	71	:5F
7280	FF	1C	00	01	EE	FE	EF	00	:F7
7288	03	FB	FF	BF	80	07	F1	FF	:33
7290	1F	C0	0F	E0	FE	0F	E0	0F	:CA
7298	E0	FE	0F	E0	0F	E4	FE	4F	:0D
72A0	E0	0F	F1	FF	1F	E0	0F	FF	:EC
72A8	FF	FF	E0	01	C0	38	07	00	:DE
72B0	00	80	10	02	00	00	00	00	:92
72B8	00	00	00	00	00	00	00	00	:00
72C0	00	00	00	00	00	00	7F	00	:7F
72C8	00	00	07	FF	F0	00	00	38	:2E
72D0	FF	8E	00	00	F7	7F	77	80	:FA
72D8	01	FD	FF	DF	C0	03	F8	FF	:96
72E0	8F	E0	07	F0	7F	07	F0	07	:E3
72E8	F0	7F	07	F0	07	F2	7F	27	:05
72F0	F0	07	F8	FF	8F	F0	07	FF	:73
72F8	FF	FF	F0	00	E0	1C	03	80	:6D
7300	00	40	08	01	00	00	00	00	:49
7308	00	00	00	00	00	00	00	00	:00
7310	00	00	00	00	00	00	3F	80	:BF
7318	00	00	03	FF	F8	00	00	1C	:16
7320	7F	C7	00	00	7B	BF	BB	C0	:FB
7328	00	FE	FF	EF	E0	01	FC	7F	:48
7330	C7	F0	03	F8	3F	83	F8	03	:6F
7338	F8	3F	83	F8	03	F9	3F	93	:80
7340	F8	03	FC	7F	C7	F8	03	FF	:37
7348	FF	FF	F8	00	70	0E	01	C0	:35
7350	00	20	04	00	80	00	00	00	:A4
7358	00	00	00	00	00	00	00	00	:00
7360	00	00	00	00	00	00	1F	C0	:DF
7368	00	00	01	FF	FC	00	00	0E	:0A
7370	3F	E3	80	00	3D	DF	DD	E0	:7B
7378	00	7F	7F	F7	F0	00	FE	3F	:22
7380	E3	F8	01	FC	1F	C1	FC	01	:B5
7388	FC	1F	C1	FC	01	FC	9F	C9	:3D
7390	FC	01	FE	3F	E3	FC	01	FF	:19
7398	FF	FF	FC	00	38	07	00	E0	:19
73A0	00	10	02	00	40	00	00	00	:52
73A8	00	00	00	00	00	00	00	00	:00
73B0	00	00	00	00	00	00	0F	E0	:EF
73B8	00	00	00	FF	FE	00	00	07	:04
73C0	1F	F1	C0	00	1E	EF	EE	F0	:BB
73C8	00	3F	BF	FB	F8	00	7F	1F	:8F
73D0	F1	FC	00	FE	0F	E0	FE	00	:D8
73D8	FE	0F	E0	FE	00	FE	4F	E4	:1C
73E0	FE	00	FF	1F	F1	FE	00	FF	:0A
73E8	FF	FF	FE	00	1C	03	80	70	:0B
73F0	00	08	01	00	20	00	00	00	:29
73F8	00	00	00	00	00	00	00	00	:00

デモプログラム(モンスター移動プログラム)の使い方

1．CLEAR,&H6FFF を行い，マシン語領域を確保します。

2．モンスター移動プログラムを入力します（アセンブルリストにはモンスターのドットデータは含まれていません。ダンプリストどおりに入力してください)。

3．転送実行プログラムを入力しRUNします（モンスター移動プログラムのバイト数は1019です)。

結果として赤と緑の２匹のモンスターが画面左端から右にむかって２：３の速度比で移動します。

画面制御プログラムを作る際の注意

このプログラムを使えばBIOSとかメイン，サブ間のデータ受け渡しなどという面倒なことを一切考えずに，画面制御プログラムを組むことができます。が，いくつかの注意点があります。

1 画面制御プログラムはメインの＄7000以降に４Kバイト以内で作りますが，いかにも＄C000番地以降にあって直接VRAMをアクセスする感覚で作ってください。

2 プログラムの先頭には必ずLDA，＄D409（マシンコードでB6D409）の３バイトをつけてください。これはVRAMアクセスフラグのONで，FMシリーズでVRAMを直接制御するときにはぜひ必要です。なくても表示はできますが，画面が乱れます。

3 画面制御プログラムはサブの＄C000を開始番地として実行されるので，作るときはそのことを十分意識してください。アセンブラなどでプログラムを開発する場合，エクステンドモードは注意して使う必要があります。

・VRAMのアドレス以外はエクステンドモードを避けポジションインディペンデントに作ればいいわけです。そのもっとも簡単な方法はプログラムカウンタ相対オフセットの多用です。

4 画面制御プログラムは必ずメインの＄7000を開始番地として４Kバイト以内で作ってください。その際，サブルーチンからの戻りは必ずRTS命令で行ってください。またCLEAR文の設定は自分で行ってください。

モンスター移動プログラムのアルゴリズム

あるドットパターンを横に８ドットごと動かしたり，縦に１ドットごと動かすのは簡単ですが，横に１ドットごと動かすのはVRAMの構成が横８ドットを１バイトで管理するようになっているため，少々やっかいです。そこで，それを実現する簡単な方法を紹介します。

まず，横８ドット分におさまる**図２**のようなドットパターンを作るとします。

この右側にさらに８ドット分の空白を付け加えて**図３**のパターン０を作ります。次にパターン０のドットパターンを右に１ドット分シフトしたパターン１を作ります。同様にしてパターン２～パターン８を作り，合計９個のパターンを用意します。

もうおわかりでしょう。VRAMのn，n＋1番地にまずパターン０を書き込み（初期状態)，次はパターン１，その次はパターン２……という具合にパターン８まで順番に書き込んでいけば，いかにも絵が動いているように見えるはずです。

図２

図３

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
パターン０ …	●	●	●													
パターン１ …		●	●	●												
パターン２ …			●	●	●											
パターン３ …				●	●	●										
パターン４ …					●	●	●									
パターン５ …						●	●	●								
パターン６ …							●	●	●							
パターン７ …								●	●	●						
パターン８ …									●	●	●					

パターン８まで表示したら次はVRAMの番地を１つ増やし，n＋1，n＋2番地に今度はパターン１からパターン８まで書き込みます。そしてまた番地を１つ増やして……，というようにくり返せば絵を１ドットごとに横に移動できます。パターン０は最初に１度使うだけです。

この方法はスピードの点ではもっとも速いと思われますが，パターンをたくさん用意するのでメモリを食うのが難点です。

さてモンスター移動プログラムではモンスターのドットパターンを横40ドット，縦16ドットの計80バイトで構成しています（**図４**)。ドットパターン０の右端縦16バイトは，もちろんすべて０です。

モンスターの横１ドットの移動の方法はわかったと思いますので，もう１つのテクニック，２匹のモンスターを異なった速度比で移動させる方法を説明します。

赤のモンスターを１ドット分右に移動させて表示するルーチンをMONR，緑のモンスターを１ドット分右に移動させて表示するルーチンをMONGとします。赤と緑のモンスターを２：３の速度比で移動させるには，サブルーチンMONRを２回コールする間に，サブルーチンMONGを３回コールすればよいのです。これを，

```
BSR   MONG
BSR   MONR
BSR   MONG
BSR   MONR
```

図4　モンスターのドットパターン0

BSR　MONG

と記述してしまうと，速度比を変更する場合複雑になり，たとえば赤：緑＝7：8などというとBSR MONRを7回，BSR MONGを8回書かねばなりません。そこで，サブルーチンMONR，MONGの中にカウンタを設ける方法をとります。まず，メインループでは必ずMONRとMONGを1回ずつコールすることにします。

メインループ　BSR　MONR
　　　　　　　BSR　MONG

そしてサブルーチンMONRもMONGもコールされるごとにカウンタを1減らし，カウンタ値が0でないときは何もせずリターン，カウンタ値が0のとき初めてモンスターが1ドット分右に移動します。カウンタはMONRでは初期値を3，MONGでは初期値を2にしておくと，MONRを3回コールして赤のモンスターが右に1ドット，MONGを2回コールして緑のモンスターが右に1ドット移動します。

このようにするとメインループを6回まわると，MONRは2回だけ赤のモンスターを移動させ，MONGは3回だけ緑のモンスターを移動させますから，2：3の速度比が得られます。

プログラムはトップダウン方式（最初にメインルーチンを作成し，順次下位のサブルーチンを作っていく方法）で作りました。詳細はアセンブルリストを参照してください。

最後に

すでにマシン語に精通していらっしゃる方にはわかりきったことかもしれません。しかし，私がマシン語をマスターするまでの長い道のりを思いだし，マシン語学習に少しでも役に立てばと思います。先述のようにマシン語の勉強には画面上で絵を動すようなプログラムを作るのが最良です。単なる計算プログラムでは興味が持続しません。画面制御はロード，ストアのくり返しですから，すぐに自分でプログラムが作れるようになるでしょう。また，疑問がわいたらすぐ機械を前に実験してみることです。FMではほかのマシンと違って，入門者が画面制御からはいるのは，ちょっとたいへんなのですが，本プログラムを使えばPC，LEVEL-3の感覚で直接VRAMを制御するのとかわりありません。むしろVRAMの構成がすっきりしているFMの方が画面制御はやりやすいと思います。

マシン語はBASICと比べれば確かに不便ですが，その速度，メモリ容量の少なさはBASICの比ではありません。また，アセンブラなどの開発ツールを使えばプログラム開発の労力もぐっと減ります。

マシン語を勉強している人の一助になれば幸いです。

F-BASIC中級入門

3

桑原　岳夫

はじめに

創刊号のアンケートはがきの集計結果を見る機会がありました。それによると，読者の関心は機械語に向いているという結果が出ていました。マイコンとの出会いがパーソナルコンピュータであるという現在では，機械語に対して非常な期待をもつのも無理からぬことだと思います。一般にBASICで記述されたゲームソフトは，速度の面でイライラさせられることが多く（リラックスするためのゲームでイライラするのは精神衛生上非常に良くない），市販のゲームソフトの素早い動きを見ると，自分で作ってみたいと思うのも無理からぬことです。

しかしマイコンのモニタを使ってプログラムを開発しようとすると，必ずといっていいほど挫折すること請け合いです。とくに68系のコンピュータは，分岐命令に相対分岐を使用しているために，アドレス計算に計算機（マイコンがとてもありがたく感じます）と指の助けを借りる必要がでてきます。そして作成されたプログラムが動作するか否かは，まさしく動かしてみなければわからず，多くの場合は暴走します。完成（もちろん動作した時点を完成と考えるのですが）するまでには，多くの時間と集中力，くやしい思いをかみしめることとなります。もちろん，プログラマにも天才肌の人間（数キロバイトのプログラムを一発で完動させることができれば，まぎれもない天才です）や，メモリのダンプリストを見てプログラムを理解していく職人肌の人間などがいます。

しかしこれらの人々のまねをすれば，自分の馬鹿さかげんをしっかりかみしめるのが落ちです。まさしくコンピュータは〝自分を見つめるための鏡〟であることがよくわかります。絵の世界でも将棋の世界でも，頂点を極めることのできるのはほんの一握りの人で，多くの人達はそれを見て楽しむのが関の山です。だからといって，その行為が無駄ではなく，趣味なら自分が楽しめばそれで良いと思います。

話が脱線しましたが，結局いいたいことは自分の能力に自信がもてないのなら，秀れた道具を手に入れるのが一番だということです。機械語を開発するのなら，最低でもアセンブラ，ディスアセンブラ，強力なモニタおよびフロッピーディスクは必需品でしょう。現在でもこれだけそろえるとなると20万円前後必要ですが，これだけあればかなりのことができると思います。一番大切なのは，熱意であることは言うまでもありません。ちなみにプロと呼ばれる人は，特殊な場合を除いて，機械語は使わず高級言語を使う傾向にあるようです。コンピュータが使われ始めた初期の段階（たかだか十数年前ですが）では，その能力が貧弱であるために，機械語を使うことが多く，また使わねば仕事になりませんでしたが，計算機の能力が格段に向上した現在では，大切なのはソフトウェアのメンテナンス（保守）であり，ドキュメンテーション（文書化）です。BASICでもREM文が用意されていますが，これを有効に使っている人は多くない様です。でも，例えば，数値計算の解説書などを見ると，プログラムよりもコメントの方が多いなどということはざらにあります。

人間は忘却することが当たり前であり，長いプログラムを作るときなどは作りながら忘れていくことがあります。短いプログラムならいざ知らず，長い場合はコメントをかなり有効に使うべきでしょう。マイコンの能力が貧弱であり，コメントを挿入すると実行速度が落ちるというインタプリタ系言語の欠点は問題ですが，もうしばらくすると，機械語を使用しなくても十分な速度が得られる様になるはずです。プロになりたいのなら別ですが，機械語相手に苦労する前にこれらのソフトウェア技法を，BASICをたたき台にして訓練しておくほうがためになると思います。これらのことは，“言うはやすし行うは難し”で，筆者自身BASICのプログラムを書き下していき，なんとか動く様につじつまを合わせる方が得意です。こんな習慣を身につけると，構成力を要求されるプログラムや他人と共同で作成するプログラムに対して，極端に能率が落ちてしまいます。これまでの経験では，紙のうえで充分プログラムを作り，デバッグを行ったうえ，コメントを充分書き込んだプログラムを作成するのが，遠回りのようでいて一番の近道だと思われます。良いコメントを書くというのは大変なことです。

適当なFORTRANの解説書により，コメントの書き方を勉強してみてください。

ランダムファイルについて

前置きが長くなりましたが，いよいよランダムファイルに話を移します。

ランダムファイルはシーケンシャルファイルと異なり，ファイルの内容についてはユーザーの管理にゆだねられています。F-BASICのファイルは，256バイト固定長であるため，その使用方法については考慮の余地があります。

FIELD文ではバッファの内容を最低２つに分割する必要があります。それは文字変数が最大255文字しか収納できないために，256文字分の大きさのバッファは１変数で代用できません。機械の内部で文字変数は文字の長さとそれの先頭アドレスをそれぞれ１バイト，２バイトで表し，変数が空である時に長さの部分に０を収納しています。そのために255文字以上は表現できないのです。文字変数の収納部の大きさは，ユーザーの方で自由に設定できますが，これはCLEAR命令で行います。初期値として300バイト分が確保してありますので，自由に定義できます。文字変数は使うたびに収納部の先頭から変数をつめていき，収納部がいっぱいになると実行を中止して，ちり集め（ガベージコレクションと呼びます）を行います。これは収納部を文字変数のポインタを見て，使われていない部分を作り出します。これに対してFIELD文で指定した文字数のポインタは，バッファ（ファイル内）を指しています。

このために両者の間で演算を行うことは変数管理上好ましくありません（というより不可能です）。ただし置き換え，定義は可能であり，読み出すのは一向に構いません。これらはLSET，RSET，MID命令であり，通常のLET，MID命令の様なものです。

確かにこれは必要な作業ですが，わずらわしいことです。そのために，普通はバッファから変数を通常変数にコピー（といっても代入命令で）し，加工を行った後にSET命令でバッファに代入する形をとっているようです。こうすることにより自由に変数を使用できます。

F-BASICは文字変数に対する処理がしやすく（MICRO SOFT社のBASICはだいたいそうですが），有効に利用すると複雑な処理を簡潔に行えます。

ランダムファイルにおいては，数値も文字変数として扱わねばならないために，変数の扱いに慣れることは大切なことで，ついでに変数の内部表現にも理解をもっておかねばなりません。

F-BASICでサポートしている変数の型は，整数，実数，倍精度実数および文字変数の４種です。ここで整数は２バイト（文字にして２文字分）の大きさで表現されています。整数は正負をもっており，その符号に最上位ビットを用いています。ここで使われているのは２の補数という考え方です。ためしにいろいろな整数を16進数（HEX$命令を用いれば良い）で表してください。0～7FFFまでの16進数が正で，8000～FFFFが負数を表しています。16進数の用途は，一般には少ないのですが，機械の内部をモニタで見るときには必要です。なぜこの方法を使うのかというと，ひき算をたし算で表現できるからです。実数の表現はこれと異なり，浮動小数点数を使っています。浮動小数点は数値を指数で表現し，記憶するときに有効数字を表す仮数部とセットにして表すものです。**図１**に，その例を示します。

ここで通常の実数は指数部に１バイト，仮数部に３バイト，倍精度では指数部に１バイト，仮数部に７バイトを要します。内部とのかかわりは文法書のUSR関数の項を参照してください。

ランダムファイルを管理するためにユーザーに開放されている手段は，レコード番号のみです。目的のデータがどのファイルに収納されているかを調べるにはどうしたら能率が良いかを考えることは重要です。このファイルの名札の様なものをキー（KEY）と呼び，これを用いて目的物を探す作業を検索といいます。ランダムファイルの性格上，この作業は必ず使います。このためにいかに早くキーから目的のデータを検索できるかが，ソフトウェアの良否を決定します。しかし実際には，完成されたファイルのレコード参照をすることにより，そのファイルに挿入，削除，ファイル変更を行うことが一般的な要求です。この場合，シーケンシャルファイルは全く不適当です。もちろん不可能ではないにしろ，データの移動量が大きくなるために，コンピュータの内部メモリで処理できない大きさのファイルでは，非実用的です。そのために，ファイル管理には一般にリンク付きリストが用いられています。

この構成を**図２**に示します。

つまりレコードは，そのレコードのもつキーと，次につながるレコードをもつポインタを持っており，レコードの物理的な位置（レコード番号のことです）と無関係に，有機的に結合しています。定義すれば，シリアルなファイルを持つリストであり，シーケンシャルファイルと異なり連続したロケーションに置かれず，それぞれのキーに

図１

S　指数部　仮数部
符号
1　隠れた最上位ビット

図２

よって決まる関係に基づき，次のレコードのロケーションを示すポインタを持つ，ということです。

これを基本に据え，ポインタを多数持ったり，キーを集めてファイルを作ったディレクトリファイルなどのサブリストを持ったりするものがあります。シーケンシャルファイルを組み合わせれば，また異なる応用もできます。

この考えをとりいれて作られたＯＳが，FLEX(68系のFDOS)やOS-9(6809のOS)です。F-BASICやCP/M（80系のFDOS）はキーとポインタを別に設置した形式をとっています。文法書に詳しい内容が記してありますので参照してください。ちなみに前者をディレクトリ，後者をFAT(ファイルアロケーションテーブル）と呼んでいます。どちらの方式が優れているかは一概に言えず，長所も短所もあるのですが，応用が効くという点で言えば，前者が優れています。要は使い分けの問題だと思います。

詳しく知りたい方は，良い参考書が多数出ていますので御一読をおすすめします。

おわりに

はじめにも書きましたが，ユーザーが機械語という低級言語を使わなければならないのはあきらかに異常(?)で，一種の退化だと思います。もうしばらくすると機械語を使わなくともよくなるようになると思いますので，不評のBASICですが自分のやり方でかわいがってください。これほどいい加減で使いやすい言語もあまりないのですから。

次回は，ファイルを利用したセミデータベースもどきの住所録を作る予定です。

虫つくろいのページ
（第2号の訂正）

赤色の部分が訂正事項です。

●P.19　表1

①CPU

……メインCPUと~~I/O処理用の~~サブCPUが独立

（CRT，キーボード処理はサブCPU，プリンタ，フロッピーはメインCPUが処理します）

⑦カセットインタフェース

……，~~半速サッポロ方式~~富士通方式

（詳しくは，次号記事「テープの読み取りミスを解決！」に掲載します）

●P.26　左段3行目　　P.27　左段14行目

…I/OやBIOS，サブシステムなど全て公開してくれて…。

…「BIOS＆ブートストラップローダ解説書」なる立派なマニュアルが本体に付属してくる。

➡BIOS，サブシステムは公開していません。

●P.31　中段5行目

…FM-16(micro16s）が~~4~~5台，FM-7が~~2~~1台

●P.78

3512 IF GN(N)=1 GOTO ~~6020~~ 6030

●P.85　表1

LFD-550/FM	㈱東京電子科学機材	〈オプション FDインタフェースカード ~~33,000円~~ 15,000円	バスドライバに付属	W 148,000 合計 ~~181,000~~ 163,000

●P.97　前文右段12行目

CLEAR300，&H5FFF

●P.105　図1

FIRQ→……のみプッシュ：E=~~1~~0，I=1，F=1に……

●P.106　3Dグラフィックス実行方法補足説明

第2号に掲載された田中明雄氏の三次元グラフィックスのプログラムについて，実行方法がわからないという質問をいただいたので，ここに実行方法を説明しよう。

このプログラムはFM-7, 8, 11, 各機種で共通に動くが，実行の前に，FM-7ではSCREEN　7，7が，11ではSCREEN　0がそれぞれ実行されていなければいけない。その後は各機種共通に次の手順を行う。

①CLEAR，&H3FFF⏎

（⏎はリターンキーの意味である）

②第2号109〜112ページ，または右の**リスト**のとおりに機械語を入力する。モニタで入力してもよいし，**リスト**を見ながら今号掲載のチェックサムプログラムを使って入力するのもよいだろう。＄4000番地から＄4121番地までを入力する。

Add	+0	+1	+2	+3	+4	+5	+6	+7	Sum
4000	8D	49	86	A6	B7	50	91	86	:20
4008	A6	B7	50	90	8D	60	B6	50	:30
4010	90	8B	01	B7	50	90	81	5A	:8E
4018	2F	F2	B6	50	91	8B	05	B7	:FF
4020	50	91	81	5A	2F	E1	86	A6	:F8
4028	B7	50	90	86	A6	B7	50	91	:5B
4030	8D	3C	B6	50	91	8B	01	B7	:A3
4038	50	91	81	5A	2F	F2	B6	50	:E3
4040	90	8B	05	B7	50	90	81	5A	:92
4048	2F	E1	39	8E	50	00	86	10	:BD
4050	A7	80	4F	A7	80	CC	50	08	:C1
4058	ED	81	CC	00	7B	ED	81	8E	:B1
4060	50	0A	86	17	A7	80	86	14	:B8
4068	A7	80	BF	50	88	39	8D	09	:8D
4070	8D	23	8D	42	8D	55	8D	6F	:5D
4078	39	B6	50	90	17	00	99	1E	:9D
4080	89	B6	50	91	17	00	91	3D	:05
4088	48	1E	89	1D	C3	00	80	44	:93
4090	56	FD	50	92	39	FC	50	92	:4C
4098	10	83	00	60	2F	06	86	07	:B5
40A0	B7	50	8E	39	10	83	00	49	:AA
40A8	2F	06	86	03	B7	50	8E	39	:8C
40B0	86	02	B7	50	8E	39	F6	50	:9C
40B8	90	1D	FD	50	94	F6	50	91	:65
40C0	1D	B3	50	94	C3	01	40	FD	:B5
40C8	50	8A	39	F6	50	90	1D	FD	:03
40D0	50	94	F6	50	91	1D	F3	50	:1B
40D8	94	47	56	47	56	B3	50	92	:63
40E0	C3	00	96	FD	50	8C	39	BE	:29
40E8	50	88	FC	50	8A	ED	81	FC	:18
40F0	50	8C	ED	81	B6	50	8E	A7	:85
40F8	80	4F	A7	80	BF	50	88	7A	:07
4100	50	8F	27	01	39	86	14	B7	:91
4108	50	8F	CC	50	0C	FD	50	88	:DC
4110	8E	50	00	AD	9F	FB	FA	39	:58
4118	8E	50	96	4D	2C	01	40	A6	:D4
4120	86	39	00	00	00	00	00	00	:BF

③第2号112ページ下に掲載の**BASICプログラム**を入力する。

④RUN⏎

⑤EXEC &H4000⏎

以上の手順で実行できる。意外な速さに驚かされるだろう。

このプログラムのセーブはプログラム名を3Dとすれば，

SAVEM"3D", &H4000, &H50FF, &H4000⏎

としておけばよい。こうしてセーブしておくと以後の実行は

LOADM"3D",'R⏎

で可能で，さらに実行する場合は，

EXEC⏎でよい。

ゲームリスト入力のテクニック

小林　安男

ゲームリストを入力したが動かない，近くに教えてくれる友人もいない，どうにかして……と悩んでいる仲間も多いと思います。

ここでは，ゲームリスト，特にBASICのリストに絞って，入力上の注意点などを解説してみましょう。

まず，間違えやすい文字・記号を示します。

O	オー	0	ゼロ
I	アイ	1	イチ
B	ビー	8	ハチ
C	シー	G	ジー
,	カンマ	.	ピリオド
;	セミコロン	:	コロン
−	マイナス	−	カナ¥

O(オー)と0(ゼロ)

オーとゼロは，ゼロがØの字体のときは間違えないと思います（Øを使っても，間違える人が結構います)。

斜線の出ないゼロのときは，字体をよく比べて区別してください。

よく使われる BASIC の命令をあげると

```
FOR      TO      GOSUB
CONSOLE  LOCATE  COLOR
GOTO     POKE
```

などがあります。前後の文字により判断します。

I(アイ)と1(イチ)

印刷の具合により，よく間違えやすいものです。アイの入る主な命令をあげると，

```
IF   DIM     INKEY$   WIDTH
LINE PRINT   PRINT USING
LINEINPUT    INPUT  INPUT$
```

などがあります。

数字は，アルファベットの後に変数の添字として使われることもあるので，注意が必要です。

B（ビー）と8（ハチ），C（シー）とG（ジー）なども，見間違えることがあるので

```
I0  I1  I2  ....
B0  B1  B2  ....  B8  B9
O0  O1  O2  ....
```

要注意です。

※　※　※

次に記号で間違えやすい例をあげてみましょう。

.	ピリオド	,	カンマ
:	コロン	;	セミコロン

.(ピリオド)

これは，数値の小数点表示やダブルコーテーションの中で使われるだけです。リストを打ち込むときに，省略形でする場合にはよく使うキーです。また，BASIC などで0.5の表記は単に .5と書きますので，注意してください。

,(カンマ)

よく出てきます。

```
10 DATA 8,4,0,6
20 WIDTH 40,25
30 LINE (5,5)-(34,19)
40 LOCATE 20,2
50 COLOR 7,0
```

行番号10はデータ文の区切りに，行番号20～50はパラメータの区切りに使われています。

とくにデータ文では，データの数値に小数点としてピリオドが使われていることもあり，注意が必要です。ピリオドとカンマを間違えると，Out of Data エラーになったり，間違ったデータが読み込まれたりします。READ 文などでエラーが出たときは，DATA 文を二人で「読み合わせ」するとよいでしょう。

一人は雑誌などのリストを声に出して読み上げます。CRT は見ずに，元のリストのみ追っていきます。もう一人は CRT 上のリストをカーソルキーなどを使って追っていきます。

:(コロン)

マルチステートメントの区切りに使われています。

```
10 BEEP:GOTO 280
20 CLS:LOCATE 7,18
30 PRINT "83.05.18"
```

のように使っています。

;(セミコロン)

セミコロンは，PRINT 文や INPUT 文の後に使われ，次に表示する文字を続けて出すときに使われます。

```
10 CLS
20 INPUT "NUMBER=";A
30 PRINT " Oh! FM ";
40 PRINT "VOL";A
50 END
```

−(マイナス)と−(音引)

LINE (3, 6)−(36, 11)

TX＝TX−2

などに使われている−(マイナス記号)は右のテンキーの中と，[−= ホ]のキーを使います。

カタカナの長音（[¥ | _]のカナモード）はマイナス記号と似ていますが，コンピュータの内部処理では，全く別の働きをします。

普通，BASIC 文法中では，すべてマイナスを使います。カタカナの長音(音引)はプリント文で，

PRINT “キーニュウリョク”

のように，“ ”で囲まれた文字列の中で使われるだけです。

※　※　※

以上の注意点に留意しながら入力し，デバッグのときもこれらを重点的に行うとよいでしょう。

リスト入力のデバッグの最良の方法は，ただ一つ，BASIC の文法を理解することです。わからないコマンドに出会ったときはBASIC 文法書を広げ，よく読んでください。

こまめに読んでください。忘れてしまってもかまいません。繰り返し繰り返し読むと理解が深まります。ただ入力するだけでなく，なにをパソコンにさせているのか，1行1行理解することが大切です。やがては自分でゲームを作ることもできるようになります。がんばってください。

for FM-7,8,11 ………ゲームリスト………

CSK「B-1核爆撃機」

あなたはB-1核爆撃機隊の指揮官だ。指示されたソビエト内の目標都市を核弾頭つきの短距離ミサイル（SRAM）で爆撃し，スール空軍基地に帰還できるか。ソビエト軍（コンピュータ）は無数のMIG戦闘機や地対空ミサイル（SAM）で攻撃してくる。あなたに与えられた武器は6発のフェニックスミサイルと，敵よりすぐれた電子兵器（ECM）だけだ。

プログラムをRUNさせると画面に目標都市，セーフティコード，敵基地などが表示される。必要な部分はメモしておくとよいだろう。

コマンドの説明

フライトコントロール・コマンド

AL（高度設定）：高度は100～25000mの範囲で自由に決められる。

CO（コース設定）：0～360°の範囲で入力。

AU（自動操縦）：時，分，秒を入力するとその時間は現在の高度，コースを維持する。途中で敵の攻撃を探知するとその地点で解除される。

航法コマンド

NA（航法）：指示した都市までのコースと距離を表示する。

SE（サーチ）：現在のコースの左右45°の範囲の最も近いソビエト基地の状態とそこまでの距離を調べる。かなり時間を要するコマンド。

ST（状況報告）：現在のB-1の状況を表示。

RA（レーダー）：B-1に向かってきているMIG，SAMとコンタクトまでの時間を表示する。

コンバット・コマンド

EC（ECM）：SAMを自爆させたりMIGにこちらを見失わさせる。連続使用すると効果が弱まる。

EV（緊急回避）：高度とコースを急激に変化させる。コンタクトまで数秒以内の敵に対し有効だが，低空で行うと墜落の恐れがある。

PH（フェニックスミサイル）：射程は200km

```
モクヒョウ トシ SVERDLOVSK ヲ バクゲキシテ、
THULE キチニ キカンセヨ。
アンゼンソウチノ カイジョ コード ハ

モクヒョウトシ ノ コウゲキガ フカノウナ トキハ、ツギノ イズレカノ
 トシヲ コウゲキセヨ。

 ARKHANGELSK ASTRAKHAN    KIYEV
 LENINGRAD   MOSKVA       MURMANSK
 SEVASTOPOL  VOLGOGRAD    YEREVAN

SOVIET ノ ボウエイ キチ ハ、ツギノ トオリ
 DUBOVKA      KHARKOV     KONOSHA
 LIPETSK      OCHAMCHIRA  ODESSA
 ONEGA        PECHENGA    PINSK
 PODOLSK      PSKOV       ROSTOV
 DARPA        SYKTYVKAR   TALLINN
 BILISI       UKHTA       VINNITSA
 VYBORG       YARANSK

GOOD LUCK!

COMMAND?
```

指令が下った。戦闘開始。セーフティコードはメモしておこう。

```
 PHOENIX ミサイル ノコリ 6

 モクヒョウ トシ: ASTRAKHAN

COMMAND? NA

WHERE TO? AS
ASTRAKHAN   ホウイ 151.3 ° キョリ  4603 km.

COMMAND? CO

NEW COURSE? 151.3

COMMAND? AU

TIME FOR AUTOPILOT (時,分,秒) ? 5,0,0

PECHENGA キチカラ、SAM- 1 ガ ムカッテクル!

 SAM-  1 INTERSEPTS IN  457 SECONDS.

COMMAND?
```

目標都市のASTRAKHANまで4603km。まだまだ遠い。うわ，SAMが攻めてきた。

```
TARGET? ON
PHOENIX MISSILE ハッシャ !!
ONEGA キチニ メイチュウ。 キチハ、シヨウ フノウ。
 SAM-  9 INTERSEPTS IN   49 SECONDS.
 SAM-  4 INTERSEPTS IN   93 SECONDS.
 SAM-  4 INTERSEPTS IN   72 SECONDS.

COMMAND? EV

 SAM-  9 INTERSEPTS IN   36 SECONDS.
 SAM-  4 INTERSEPTS IN   80 SECONDS.
 SAM-  4 INTERSEPTS IN   59 SECONDS.

COMMAND? EV

SAM- 4 ハ、モクヒョウヲ ミウシナッタ.
 SAM-  9 INTERSEPTS IN   21 SECONDS.
 SAM-  4 INTERSEPTS IN   44 SECONDS.

COMMAND? EC

SAM- 9  ハ、ジバクシタ。
SAM- 4  ハ、ジバクシタ。

COMMAND?
```

ONEGA基地が射程距離にはいった。ミサイル発射，命中／しかしSAMが攻めてくる。EVとECで回避しよう。

```
COMMAND? CO

NEW COURSE? 161.2
 SAM-  9 INTERSEPTS IN  198 SECONDS.
 MIG- 31 INTERSEPTS IN   59 SECONDS.
 MIG- 29 INTERSEPTS IN  322 SECONDS.
 MIG- 33 INTERSEPTS IN  871 SECONDS.

COMMAND? NA

WHERE TO? AS
ASTRAKHAN   ホウイ 162.2 ° キョリ  1731 km.

    NUCLEAR AIRBURST!!!

B-1 ハ、ヤマノ チュウフクニ ゲキトツシタ。

サクセン レポート  :
ボウエイ キチ ハカイ   :
    KONOSHA
    ONEGA
ノリクミイン ゼンイン シボウ。

        TRY AGAIN (Y/N)?
```

山の中腹に激突，全員死亡。南無阿弥陀仏……。

で，指示した目標に発射される。目標はMIG，SAMのどちらか，または敵基地の頭2文字を入力する。

爆撃コマンド

AR（安全装置解除）：このコマンドを入力するとセーフティコードの入力を要求してくるので，ゲーム開始時に与えられたコードを入力する。

BO（SRAM発射）：目標を入力すると最終決定を聞いてくるのでYかNで答える。Yを入力するとSRAMが発射される。


```
10 WIDTH 40,25:CONSOLE 0,25,0,0:LOCATE 0,0,0
20 RANDOMIZE (TIME-INT(TIME/65536!)*65536!-32768!):N0=10:N1=20:N2=N0+N1
30 DIM C$(24),C(10,2),N$(N2),T(N2,3),M$(1)
40 PRINT TAB(9);"** B-1 BOMBER GAME **":PRINT
50 PRINT TAB(5);"ORIGINAL COPYRIGHT  1980 BY":PRINT:PRINT TAB(13);"AVALON HILL"
60 PRINT TAB(9);"MICROCOMPUTER GAMES"
70 PRINT:PRINT:PRINT
80 PRINT TAB(7);"COPYRIGHT IN JAPANESE 1982"
90 PRINT TAB(5);"KIYA Overseas Industry Co. Ltd."
100 FOR I=1 TO 24:READ C$(I):NEXT
110 DATA au,AU,ec,EC,ev,EV,ph,PH,na,NA,al,AL,co,CO,st,ST,ra,RA,se,SE,ar,AR,bo,BO
120 FOR I=1 TO 1000:NEXT
130 FOR I=1 TO N2:READ N$(I),T(I,1),T(I,2):NEXT
140 N$(0)="THULE  AFB":T(0,1)=0:T(0,2)=0
150 DATA ARKHANGELSK,8261,5922,ASTRAKHAN,8631,7947,KIYEV,7391,7467
160 DATA LENINGRAD,7661,6427,MOSKVA,7956,6937,MURMANSK,8056,5452
170 DATA SEVASTOPOL,7506,8152,SVERDLOVSK,9311,6682,VOLGOGRAD,8331,7692
180 DATA YEREVAN,8381,8657
190 DATA DUBOVKA,8350,7650,KHARKOV,8850,7450,KONOSHA,8153,6325
200 DATA LIPETSK,8100,7210,OCHAMCHIRA,8060,8400,ODESSA,7230,8050
210 DATA ONEGA,8147,5976,PECHENGA,7976,5410,PINSK,7115,7235
220 DATA PODOLSK,7900,7010,PSKOV,7475,6780,ROSTOV,7990,8010
230 DATA DARPA,8515,7930,SYKTYVKAR,8825,6200,TALLINN,7060,6415
240 DATA BILISI,8415,8435,UKHTA,8960,5925,VINNITSA,7200,7585,VYBORG,7575,6300
250 DATA YARANSK,8575,6740
260 A9=25000:A0=100:C0=10:S=4500:P=6:F9=0:T=0:E=1.75
270 M$(0)="SAM":M$(1)="MIG"
280 X=5500+RND(1)*1500:Y=3500+RND(1)*1000
290 FOR I=1 TO N2:T(I,3)=0:NEXT:A=INT(25000*RND(1)):A1=A:R9=0
300 C=INT(RND(1)*360):C1=C:T9=INT(RND(1)*N0+1):T8=0:F=18500
310 FOR I=1 TO 10:C(I,1)=0:NEXT:F$="":FOR I=1 TO 5
320 F$=F$+CHR$(INT(RND(1)*26)+65):NEXT:L9=.3
330 CLS:PRINT "B-1 BOMBER ﾊ THULE ｷﾁ ｦ ﾊｯｼﾝｼﾀ。 "
340 PRINT "ｽﾃﾞﾆ ﾋｼﾞｮｳ ｼﾞﾀｲ ｾﾝｹﾞﾝｶﾞ､ﾊﾂﾚｲｻﾚﾃｲﾙ。 "
350 PRINT "ｾﾝｾﾝ ﾌｺｸﾄ ﾄﾞｳｼﾞﾆ､ ｷﾐﾊ B-1 ｦ ｿｳｼﾞｭｳ ｼﾃ､"
360 PRINT "ｼﾃｲｻﾚﾀ SOVIET ﾉ ﾄｼｦ ｺｳｹﾞｷｾﾖ。"
370 FOR I=1 TO 7000:NEXT
380 CLS
390 COLOR 2:FOR J=1 TO 5:LOCATE 17,1:PRINT "HOT WAR":BEEP 1:FOR I=1 TO 200:NEXT:
BEEP 0:LOCATE 17,1:PRINT "       ":FOR I=1 TO 100:NEXT:NEXT:COLOR 7
400 PRINT:PRINT "ﾓｸﾋｮｳ ﾄｼ ";N$(T9);" ｦ ﾊﾞｸｹﾞｷｼﾃ､":PRINT "THULE ｷﾁﾆ ｷｶﾝｾﾖ。"
410 PRINT "ｱﾝｾﾞﾝｿｳﾁﾉ ｶｲｼﾞｮ ｺｰﾄﾞ ﾊ ";:COLOR 2:PRINT F$:PRINT:COLOR 7
420 PRINT "ﾓｸﾋｮｳﾄｼ ﾉ ｺｳｹﾞｷｶﾞ ﾌｶﾉｳﾅ ﾄｷﾊ､ ﾂｷﾞﾉ ｲｽﾞﾚｶﾉ ﾄｼｦ ｺｳｹﾞｷｾﾖ。"
430 PRINT:J=0:FOR I=1 TO N0:IF I=T9 THEN 460
440 J=J+1:PRINT TAB(12*(J-1)+1);N$(I);:IF J<3 THEN 460
450 J=0:IF I<9 THEN PRINT
460 NEXT:PRINT:PRINT
470 PRINT "SOVIET ﾉ ﾎﾞｳｴｲ ｷﾁ ﾊ､ ﾂｷﾞﾉ ﾄｵﾘ"
480 J=0:FOR I=N0+1 TO N2:J=J+1
490 PRINT TAB(12*(J-1)+1);N$(I);:IF J<3 THEN 510
500 J=0:PRINT
510 NEXT:PRINT:PRINT:PRINT "GOOD LUCK!"
520 GOSUB 1170:PRINT:INPUT "COMMAND";A$:PRINT:IF LEN(A$)<2 THEN 560
530 CM=0
540 FOR I=1 TO 24 STEP 2:IF LEFT$(A$,2)=C$(I) OR LEFT$(A$,2)=C$(I+1) THEN CM=INT
(I/2)+1
550 NEXT:IF CM>0 THEN 690
560 PRINT "ﾕｳｺｳﾅ ｺﾏﾝﾄﾞﾊ､ﾂｷﾞﾉ ﾄｵﾘ:":PRINT "CO: ﾋｺｳ ｺｰｽ ﾍﾝｺｳ "
570 PRINT "AL: ﾋｺｳ ｺｳﾄﾞ ﾍﾝｺｳ    "
580 PRINT "ST: ﾋｺｳ ｼﾞｮｳｷｮｳ ﾚﾎﾟｰﾄ"
590 PRINT "RA: ﾚｰﾀﾞｰ ﾚﾎﾟｰﾄ"
```

```
600 PRINT "NA: ｺｳﾎｳ ｺﾝﾋﾟｭｰﾀ"
610 PRINT "SE: ﾃｷ ｷﾁﾉ ｿｳｻｸ"
620 PRINT "AU: ｼﾞﾄﾞｳ ｿｳｼﾞｭｳ ｾｯﾄ"
630 PRINT "EV: ｷﾝｷｭｳ ｶｲﾋ ﾄﾞｳｻ"
640 PRINT "EC: ECM (ﾃﾞﾝﾊﾟ ﾎﾞｳｶﾞｲ)"
650 PRINT "PH: PHONIX ﾐｻｲﾙ ﾊｯｼｬ"
660 PRINT "AR: ｶｸﾊﾞｸﾀﾞﾝ ｱﾝｾﾞﾝｿｳﾁ ｶｲｼﾞｮ"
670 PRINT "BO: ｶｸﾀﾞﾝﾄｳ SRAM ﾊｯｼｬ"
680 T0=T+3+INT(5*RND(1)):GOTO 1760
690 ON CM GOTO 800,1670,1280,1410,1010,700,730,1230,1150,1090,740,860
700 INPUT "NEW ALTITUDE";A1:IF A1>A9 THEN A1=A9
710 IF A1<A0 THEN A1=A0
720 T0=T+13+INT(15*RND(1)):GOTO 1760
730 INPUT "NEW COURSE";C1:GOTO 720
740 IF F9=2 THEN 790
750 INPUT "FAIL SAFE CODE";A$:IF A$=F$ THEN F9=1
760 IF F9=0 THEN 780
770 PRINT "BOMB ARMED.":L9=10*L9:GOTO 720
780 PRINT "WRONG CODE.":GOTO 720
790 PRINT "BOMB ALREADY DROPPED.":GOTO 720
800 INPUT "TIME FOR AUTOPILOT (時,分,秒) ";K,J,I:T2=T+I+60*J+3600*K:GOTO 1760
810 IF A$=" " THEN 850
820 I=VAL(A$):GOSUB 2520:IF A$=" " THEN 850
830 J=VAL(A$):GOSUB 2520:IF A$=" " THEN 850
840 K=VAL(A$)
850 I=-I*(I>0):J=-J*(J>0):K=-K*(K>0):T2=T+I+60*J+3600*K:GOTO 1760
860 IF F9=2 THEN 790
870 IF F9=0 THEN 980
880 INPUT "TARGET";A$
890 IF ASC(LEFT$(A$,1))>&H60 THEN A$=CHR$(ASC(LEFT$(A$,1))-32)+CHR$(ASC(MID$(A$,
2,1))-32)
900 IF LEN(A$)<2 THEN 990
910 A$=LEFT$(A$,2):GOSUB 2550:IF ((NN=-1) OR (NN>N0)) THEN 990
920 GOSUB 2540:IF RA>250 THEN 1000
930 INPUT "CONFIRM DROP (Y/N) ";A$
940 IF A$<>"Y" AND A$<>"y" THEN 720
950 FOR I=1 TO 1500:NEXT:PRINT N$(NN);" DESTROYED."
960 L9=L9/15
970 T8=NN:S=S+250:A0=A0+50:F9=2:T0=T+3+INT(5*RND(1)):GOTO 1760
980 PRINT:PRINT "ｱﾝｾﾞﾝ ｿｳﾁｶﾞ ｶｲｼﾞｮｻﾚﾃ ｲﾏｾﾝ!":GOTO 720
990 PRINT "IMPROPER TARGET.":GOTO 720
1000 PRINT N$(NN);" NOT IN RANGE.":GOTO 720
1010 PRINT:INPUT "WHERE TO";A$:IF LEN(A$)<2 THEN 1080
1020 A$=LEFT$(A$,2)
1030 IF ASC(LEFT$(A$,1))>&H60 THEN A$=CHR$(ASC(LEFT$(A$,1))-32)+CHR$(ASC(MID$(A$
,2,1))-32)
1040 GOSUB 2550:IF NN=-1 THEN 1080
1050 GOSUB 2540:GOSUB 2490
1060 PRINT N$(NN);"   ﾎｳｲ";INT(AN*10)/10;"° ｷｮﾘ ";INT(RA);"km.":PRINT
1070 T0=T+40+INT(41*RND(1)):GOTO 1760
1080 PRINT "NOT ON THE MAP.":GOTO 1070
1090 I=0:J=10000!:FOR NN=N0+1 TO N2:GOSUB 2540:IF RA>J THEN 1130
1100 IF T(NN,3)=2 THEN 1130
1110 GOSUB 2490:IF ABS(C-AN)>45 AND ABS(C-AN)<315 THEN 1130
1120 J=RA:I=NN
1130 NEXT:NN=I:IF NN>0 THEN 1050
1140 PRINT "ｶﾞｲﾄｳ ﾊﾝｲﾆ ﾃｷｷﾁ ﾅｼ":GOTO 1070
1150 PRINT:PRINT "CONTACTS:":GOSUB 1170:PRINT:PRINT:IF J=0 THEN PRINT "   NO CON
TACTS."
1160 T0=T+5+INT(RND(1)*11):GOTO 1760
1170 J=0:FOR I=1 TO 10:IF C(I,1)=0 THEN 1220
1180 J=1
1190 IF C(I,2)-T<25 THEN COLOR 2 ELSE IF C(I,2)-T<100 THEN COLOR 6
1200 PRINTUSING " & &&&## INTERSEPTS IN #### SECONDS.";M$(-(C(I,1)>10)),"-",C(I,
1),C(I,2)-T
1210 COLOR 7
1220 NEXT:RETURN
1230 COLOR 5:PRINTUSING " COURSE        ###.# T   SPEED    #### km/h";C,S
1240 PRINTUSING " ALTITUDE  ##### m    FUEL    ##### km";A,F
1250 PRINT " PHOENIX ﾐｻｲﾙ ﾉｺﾘ";P:T0=T+20+INT(RND(0)*21)
1260 IF F9<>2 THEN PRINT:PRINT " ﾓｸﾋｮｳ ﾄｼ: ";N$(T9)
```

```
1270 COLOR 7:GOTO 1760
1280 A1=200+INT(300*RND(1)):IF RND(1)<.5 THEN A1=-A1
1290 A=A+A1
1300 IF A<0 THEN 2300
1310 A1=A:C1=60+INT(60*RND(1))
1320 IF RND(1)>.5 THEN C1=-C1
1330 C=C+C1:C=C+360*(C>360)-360*(C<0)
1340 C1=C:FOR I=1 TO 10:IF C(I,1)=0 THEN 1400
1350 J=(8-2*(C(I,1)<10))/(C(I,2)-T):IF J>.95 THEN J=.95
1360 IF RND(1)>J THEN 1400
1370 PRINT M$(-(C(I,1)>10));"-";C(I,1);"ﾊ､ ﾓｸﾋｮｳｦ ﾐｳｼﾅｯﾀ";
1380 IF C(I,1)>10 THEN PRINT " ｷﾁﾆ ﾋｷｶｴｼﾀ ﾓﾖｳ";
1390 PRINT ".":C(I,1)=0
1400 NEXT:T0=T+10+INT(21*(RND(1))):GOTO 1760
1410 IF P=0 THEN 1540
1420 PRINT:PRINT "PHOENIX MISSILE ﾊｯｼｬ ｼﾞｭﾝﾋﾞﾖｼ｡":INPUT "TARGET";A$
1430 IF A$="N" OR A$="NONE" THEN 1530
1440 IF A$="M" OR A$="MIG" OR A$="S" OR A$="SAM" OR A$="m" OR A$="mig" OR A$="s"
 OR A$="sam" THEN 1570
1460 IF LEN(A$)<2 THEN 1550
1470 A$=LEFT$(A$,2)
1480 IF ASC(LEFT$(A$,1))>&H60 THEN A$=CHR$(ASC(LEFT$(A$,1))-32)+CHR$(ASC(MID$(A$
,2,1))-32):GOSUB 2550
1490 GOSUB 2550:IF NN<N0+1 THEN 1550
1500 GOSUB 2540:IF RA>200 THEN 1560
1510 PRINT "PHOENIX MISSILE ﾊｯｼｬ !!"
1520 PRINT N$(NN);" ｷﾁﾆ ﾒｲﾁｭｳ｡ ｷﾁﾊ､ｼﾖｳ ﾌﾉｳ｡":T(NN,3)=2:P=P-1
1530 T0=T+3+INT(5*RND(1)):GOTO 1760
1540 PRINT "ﾐｻｲﾙ ｶﾞ ﾅｲ ! ":GOTO 1530
1550 PRINT "ｺｳｹﾞｷ ﾌｶﾉｳ !!   ":GOTO 1530
1560 PRINT N$(NN);"ﾊ ｼｬﾃｲﾆ ﾊｲｯﾃｲﾅｲ ":GOTO 1530
1570 J=10000!:K=0:FOR I=0 TO 10:IF C(I,1)=0 THEN 1600
1580 IF C(I,2)-T>J THEN 1600
1590 J=C(I,2)-T:K=I
1600 NEXT:PRINT "PHOENIX MISSILE ﾊｯｼｬ !!":P=P-1:IF K=0 THEN 1660
1610 IF J>10 THEN 1660
1620 IF C(K,1)<10 AND RND(1)>.6 THEN 1660
1630 IF C(K,1)>10 AND RND(1)>.85 THEN 1660
1640 PRINT M$(-(C(K,1)>10));"-";C(K,1);"DESTROYED!"
1650 C(K,1)=0:GOTO 1530
1660 PRINT "MISSED!":GOTO 1530
1670 FOR I=1 TO 10:IF C(I,1)=0 THEN 1750
1680 J=E*(.5+5/(C(I,2)-T))
1690 IF J>.97 THEN J=.97
1700 IF RND(1)>J THEN 1750
1710 PRINT M$(-(C(I,1)>10));"-";C(I,1);
1720 IF C(I,1)>10 THEN PRINT " ﾊ､ﾓｸﾋｮｳｦ ﾐｳｼﾅｲ ｷﾁﾆ ﾋｷｶｴｼﾀ｡ "
1730 IF C(I,1)<10 THEN PRINT " ﾊ､ｼﾞﾊﾞｸｼﾀ｡"
1740 C(I,1)=0
1750 NEXT:E=E*(.7+.3*RND(1)):T0=T+3+INT(5*RND(1)):GOTO 1760
1760 T3=T0:I9=1:IF T0>T THEN 1800 ELSE I9=0:T3=INT(RND(1)*301)+T
1770 FOR I=1 TO 10:IF C(I,1)=0 THEN 1790 ELSE IF C(I,2)<T3 THEN T3=C(I,2)
1780 IF T>=C(I,2)-10 THEN 1790 ELSE IF C(I,2)-10<T3 THEN T3=C(I,2)-10
1790 NEXT:IF (T2<=T) OR (T3<T2) THEN 1800 ELSE T3=T2:I9=1
1800 T0=T3-T:A2=A0*T0:C2=C0*T0
1810 IF A2>ABS(A-A1) THEN A=A1
1820 IF A2<=ABS(A-A1) THEN A=A+A2*SGN(A1-A)
1830 IF C2>ABS(-360*(ABS(C-C1)>180)-ABS(C-C1)) THEN 1850
1840 C=C+C2*SGN(C1-C):C=C-360*(C<0)+360*(C>360):GOTO 1860
1850 C=C1
1860 X=X+S*T0*SIN(C*3.14/180)/3600
1870 F=F-INT(S*(T3-T)/3600)
1880 Y=Y-S*T0*COS(C*3.14/180)/3600:T=T3
1890 IF F<0 THEN 2320
1900 DE=0:CR=0:FOR I=1 TO 10:IF (C(I,1)=0) OR (DE=1) OR (CR=1) THEN 1970ELSE IF
C(I,2)>T THEN 1960
1910 COLOR 2:PRINT "     NUCLEAR AIRBURST!!!":R9=1:C(I,1)=0:I9=1:COLOR 7
1920 A=A+INT(3000*RND(1))*SGN(.5-RND(1)):IF A<0 THEN CR=1
1930 A1=A:F=INT(F*RND(1)):S=INT(S*(.5+.5*RND(1)))
1940 IF RND(1)<.15 THEN DE=1
1950 GOTO 1970
```

```
1960 IF C(I,2)>T+10 THEN 1970 ELSE PRINT M$(-(C(I,1)>10));"-";C(I,1);"IN PHOENIX
 RANGE.":I9=1
1970 NEXT:IF CR=1 THEN 2300
1980 IF DE=1 THEN 2310
1990 IF F9=2 THEN 2060
2000 FOR NN=1 TO N0:GOSUB 2540:IF RA>250 THEN 2030
2010 IF T(NN,3)<>0 THEN 2030
2020 PRINT N$(NN);" ｶﾞ､ﾊﾞｸｹﾞｷ ｶﾉｳﾅ ｷｮﾘﾆ ﾊｲｯﾀ｡":T(NN,3)=1:I9=1
2030 IF T(NN,3)<>1 THEN 2050 ELSE IF RA <250 THEN 2050
2040 PRINT N$(NN);" ﾊ､ﾊﾞｸｹﾞｷ ｶﾉｳﾊﾝｲｶﾗ ﾊｽﾞﾚﾀ｡ ":T(NN,3)=0
2050 NEXT
2060 FOR NN=N0+1 TO N2:IF T(NN,3)=2 THEN 2270
2070 IF ABS(X-T(NN,1))>750 THEN 2270
2080 IF ABS(Y-T(NN,2))>750 THEN 2270
2090 GOSUB 2540:IF RA>750 THEN 2270
2100 IF RA>200 OR P=0 THEN 2140
2110 IF T(NN,3)=1 THEN 2140
2120 T(NN,3)=1
2130 PRINT N$(NN);" BASE IN PHOENIX RANGE.":I9=1
2140 IF T(NN,3)<>1 THEN 2160 ELSE IF RA<200 THEN 2160 ELSE T(NN,3)=0
2150 PRINT N$(NN);" BASE OUT OF MISSILE RANGE."
2160 IF RND(1)>L9*(T0/300)*(A^.125) THEN 2270
2170 J=0:FOR K=1 TO 10:IF C(K,1)=0 THEN J=K
2180 NEXT:IF J=0 THEN 2270
2190 IF RND(1)>.5 THEN 2210
2200 C(J,1)=INT(9*RND(1)+1):M=6500:GOTO 2220
2210 C(J,1)=21+2*INT(RND(1)*7):M=5000
2220 GOSUB 2490:L=(AN-C)*3.14/180
2230 L=SQR(M*M-(S*SIN(L))^2)+SGN(3.14-ABS(L))*ABS(S*COS(L))
2240 GOSUB 2540:C(J,2)=T+INT((3600*RA/L)+1)
2250 COLOR 6:PRINT:PRINT N$(NN);" ｷﾁｶﾗ､";M$(-(C(J,1)>10));"-";C(J,1);"ｶﾞ ﾑｶｯﾃｸﾙ!
":COLOR 7:PRINT:PRINT
2260 I9=1
2270 NEXT:NN=0:GOSUB 2490:GOSUB 2540:L=ABS(C-AN)
2280 IF (L<30 OR L>330) AND RA<6500 AND T>2000 THEN 2330
2290 IF I9=1 THEN 520 ELSE IF I9<>1 THEN 1760
2300 PRINT:PRINT "B-1 ﾊ､ ﾔﾏﾉ ﾁｭｳﾌｸﾆ ｹﾞｷﾄﾂｼﾀ｡":R9=2:GOTO 2370
2310 PRINT:PRINT "B-1 ﾊ､ ﾊﾞｸﾊﾂｼﾀ":R9=2:GOTO 2370
2320 PRINT:PRINT "  ﾈﾝﾘｮｳ ｦ ﾂｶｲﾊﾀｼﾀ｡":GOTO 2300
2330 NN=0:GOSUB 2540:IF RA<F THEN 2350
2340 PRINT "B-1 ﾊ､ ｸｳﾁｭｳ ｷｭｳﾕ ｦ ｳｹﾀ｡ ":FOR I=1 TO 1500:NEXT
2350 PRINT:PRINT "B-1 ﾊ､ THULE ｷﾁﾆ ｷｶﾝｼﾀ｡"
2360 PRINT:PRINT "ﾆﾝﾑ ｶﾝﾘｮｳ  "
2370 PRINT:PRINT "ｻｸｾﾝ ﾚﾎﾟｰﾄ :":IF T8=0 THEN 2410
2380 IF T8=T9 THEN PRINT "PRIMALY";
2390 IF T8<>T9 THEN PRINT "SECONDARY";
2400 PRINT " TARGET, ";N$(T8);" ｶｲﾒﾂ
2410 PRINT "ﾎﾞｳｴｲ ｷﾁ ﾊｶｲ  :"
2420 J=0:FOR I=N0+1 TO N2:IF T(I,3)<>2 THEN 2440
2430 PRINT TAB(4);N$(I):J=1
2440 NEXT:IF J=0 THEN PRINT TAB(4);"NONE"
2450 IF R9=1 THEN PRINT "ﾉﾘｸﾐｲﾝ ﾊ､ ﾎｳｼｬﾉｳ ﾆﾖﾘ､ ｾﾞﾝｲﾝ ｼﾎﾞｳｼﾀﾓﾖｳ｡"
2460 IF R9=2 THEN PRINT "ﾉﾘｸﾐｲﾝ ｾﾞﾝｲﾝ ｼﾎﾞｳ｡"
2470 PRINT:INPUT "          TRY AGAIN (Y/N)";A$:IF A$="Y" OR A$="y" THEN 260
2480 END
2490 DX=X-T(NN,1):DY=Y-T(NN,2)
2500 IF DY=0 THEN AN=90-180*(DX>0):RETURN
2510 AN=ATN(-DX/DY)*180/3.14-180*(DY<0)+360*(DX>0)*(DY>0):RETURN
2520 IF LEFT$(A$,1)="," THEN 1670
2530 A$=RIGHT$(A$,LEN(A$)-1):RETURN
2540 RA=SQR((X-T(NN,1))^2+(Y-T(NN,2))^2):RETURN
2550 NN=-1:FOR L=0 TO N2:IF A$=LEFT$(N$(L),2) THEN NN=L
2560 NEXT:RETURN
```

Hudoson soft「アルデバラン パート1」

レイホープ第2研究所でXRB照射実験中，ALD7ウイルスが繁殖してしまった。研究室の中にはまだ何人かの研究員がいる。ウイルスの汚染度が100%になると地球が破壊されてしまう。それを防ぐため防御壁を作るが，できるだけたくさんの人を救出したい。研究室の中に入れるのはBETA/][というロボットだけだ。ロボットをコントロールして地球を絶滅から救おう。

あそび方

画面の上方に現在の汚染度が表示される。♣はロボット，♠は研究員，■は防御壁を作るブロック。♠を右上方の隔離室に隔離する。♣はカーソル移動キー（↑↓←→）で移動する。ウイルスはどんどん広がって，道をふさいでしまうので要注意。また，♣は♠や■を1つずつ押すことしかできない。防御壁は最初に表示されたところに作らないとダメ。任務が完了したらリターンキーを押す。

ゲームルールの説明。がんばってね。

3人隔離した。残りは7人，ウイルスが広がりはじめた。

5人隔離したけれど……，全員救出できるか。汚染度は50%。残り時間は少ないゾ！

防御壁完成。結局，隔離できたのは8人だけ。リターンキーを押せ！

地球破滅はまぬがれた。ごくろうさまでした。もう1回やる人はYを，もうやめる人はNを押してね。

プログラムの説明

行番号10～30：初期設定，40：キー入力，50～110：♣移動，120～160：ウイルス増加，170～210：汚染度表示，220～260：乱数の改新，270～620：ストーリー・ルール・キー操作説明，680～700：♠と■移動，710～790：メッセージ点滅，800～820：救助した♠を数える，830～840：セットした防御壁を数える，850～870：成績表示，880～930：再ゲームするかどうか，940：♠を●(ウイルス感染)にかえる，950～990：成績判定，1010～1070：クリアしたときのメッセージ，1080～1350：画面表示，1360～1410：空白を捜して♠を10個書く，1420：壁の位置データ。


```
1 '■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
2 '■                                      ■
3 '■      ALDEBARAN PART1 L-1004          ■
4 '■                                      ■
5 '■    Copyright(C) Hudson soft          ■
6 '■                                      ■
7 '■      1981/10/1    by k.n             ■
8 '■                                      ■
9 '■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
10 COLOR7,0:WIDTH40,25:CONSOLE0,25,0
20 CLEAR300:DEFINT I:DIM IX(50),IY(50),PX(6),PY(6)
30 GOSUB1430:GOSUB220:GOSUB1080:B=1:YK=-1:TK=4
40 C=ASC(INKEY$+" "):IF C=13 THEN 800
50 XL=X-(C=28)+(C=29):YL=Y-(C=31)+(C=30):Z=SCREEN(XL,YL,0)
60 IF Z=0 OR Z=32THENIX(0)=&H420:PUT@A(X,Y)-(X,Y),IX:GOTO100
70 XS=X-2*(C=28)+2*(C=29):YS=Y-2*(C=31)+2*(C=30):P=SCREEN(XS,YS,0)
80 IF((P=0)OR(P=32))AND((Z=&HE8)OR(Z=&H87))THEN LOCATEX,Y:PRINT" ";:GOSUB680:GOT
O100
90 GOTO120
100 IFXL=0OR XL=39OR YL=2OR YL=25GOTO120
110 LOCATEXL,YL:COLOR4:PRINT"♣";:COLOR7:X=XL:Y=YL
120 L1=L1+1:M1=M1+1:L=INT(L1/150):M=INT(M1/150):IF M>14 THEN L1=150:M1=150
130 T=INT(RND(1)*2*L)+11-L:U=INT(RND(1)*2*M)+13-M
140 V=SCREEN(T,U,0):IFV=&HF0 ORV=&H87 GOTO170
150 IFV=&HE8 GOTO940
160 LOCATET,U:COLORRND(1)*6+1:PRINTCHR$(&H98+RND(1)*7.5);:COLOR7
170 YK=YK+1:LOCATE2+LK,1:COLORTK:PRINTCHR$(&H88+YK+(YK>6)*8);:COLOR7:IFYK>6THENL
K=LK+1:YK=-1
180 IFLK=19THENLOCATE2,1:PRINTSPC(19);:LK=0:YK=-1:OD=OD+10:LOCATE29,1:PRINTUSING
" ###";OD;:IFOD=100THENGOTO800
190 IF OD>=40THENTK=6
200 IF OD>=80THENTK=2
210 GOTO40
220 CLS
230 LOCATE6,10:PRINT"Do you need instruction ?"
240 FORI=0TO1:I=0:LOCATE32,10:COLOR4:PRINT"■":FORJ=0TO30:A$=INKEY$:IF A$="" THEN
 NEXTJ ELSE 260
250 LOCATE32,10:PRINT" ":FORJ=0TO30:A$=INKEY$:IF A$="" THEN NEXTJ:NEXTI
260 I=2:NEXTI:IF A$="n" OR A$="N" THEN COLOR7:RANDOMIZE TIME/4+ASC(A$):CLS:RETUR
N
270 CLS:PRINT"    ♣ ♣ ♣ ♣  ";:COLOR5:PRINT"ALDEBARAN PART 1 ";:COLOR4:PRINT" ♣ ♣
♣ ♣"
280 PRINT:COLOR7:PRINTSPC(27);"HUDSON SOFT":PRINT
290 PRINT" 2180年 4月 6日 ";:COLOR5:PRINT"ALDEBARAN ";:COLOR7:PRINT"ｹｲﾖﾘｷｶﾝ ｼﾀ NL82
1"
300 PRINT" ﾁｮｳｻｾﾝ ｶﾞﾓﾁｶｴｯﾀ ";:COLOR2:PRINT"ALD7 ";:COLOR7:PRINT"ｳｲﾙｽﾊ ";:COLOR6:
PRINT"LAYHOPE N-2":COLOR7
310 PRINT" ｹﾝｷｭｳｼﾞｮﾆﾃ ";:COLOR6:PRINT"XRB ";:COLOR7:PRINT"ｼｮｳｼｬ ｼｹﾝﾁｭｳ ｲｼﾞｮｳ ﾊﾝﾉ
ｳ"
320 PRINT" ｦ ｵｺｼ ﾄﾂｾﾞﾝ ﾊﾝｼｮｸ ｼﾊｼﾞﾒ ﾃ ｼﾏｲﾏｼﾀ｡ ｿﾉﾀﾒ"
330 PRINT" Dr.Y.HUDSON  ﾌｸﾒ 10 ﾒｲ ﾉ ｹﾝｷｭｳｲﾝ ｶﾞ ﾄﾘ"
340 PRINT" ﾉｺｻﾚ ﾋｼﾞｮｳｹｲﾎｳｶﾞ ﾊﾂﾚｲ ｻﾚ ｶﾞｲｶｲ ﾄ ｼｬﾀﾞﾝ"
350 PRINT" ｻﾚﾃ ｼﾏｲﾏｼﾀ｡ ｺﾙﾄﾞﾝ ｼﾚｲｶﾝﾊ";:COLOR4:PRINT" BETA/ｺｺ";:COLOR7:PRINT" ﾛﾎﾞｯ
ﾄ"
360 PRINT" ﾉ ｷﾝｷｭｳｼｮｳ ｦ ｹｯﾃｲ ｼ ｵｾﾝ ﾉｳﾄﾞｹｲ ｶﾞ ｻﾄﾞｳ"
370 PRINT" ｻﾚﾏｼﾀ｡ Dr.Y.HUDSON  ﾎｶ ｹﾝｷｭｳｲﾝ ｷｭｳｼﾂ ﾄ"
380 PRINT" ﾊﾝｼｮｸ ﾎﾞｳｼﾍｷ ﾌｾﾂ ﾉ ｼｷｶﾝ ｦ ｱﾅﾀﾆ ﾆﾝﾒｲｼﾏｽ"
390 PRINT" ｵｾﾝ ﾄﾞ ";:COLOR2:PRINT"100 % ";:COLOR7:PRINT"ﾆ ﾅﾙﾏｴ ﾆ ﾃﾞｷﾙｶｷﾞﾘ ｵｵｾﾞｲ
ﾉ"
400 PRINT" ｹﾝｷｭｳｲﾝ ｦ ｷｭｳｼﾂ ｶｸﾘ ｼﾃ ﾎﾞｳｷﾞｮﾍｷ 7ｶｼｮ ｦ"
410 PRINT" ｶﾝｾﾞﾝﾆ ﾌｾﾂｼﾃ ｸﾀﾞｻｲ｡";:COLOR4:PRINT" BETA/ｺｺ ";:COLOR7:PRINT"ﾊ ﾋﾄﾓ ｶﾍﾞ
```

```
ﾓ"
420 PRINT" 1ｺﾂﾞﾂｼｶ ﾊｺﾍﾞﾏｾﾝ｡ﾏﾀ ｵｽﾀﾞｹﾃﾞﾋｸｺﾄﾊ ﾃﾞｷﾏｾﾝ"
430 M$=" Hit any key to continue "
440 LINE(0,26)-(640,146),PSET,1,B
450 FORI=0TO1:I=0:LOCATE7,20:COLOR6:PRINTM$;:FORJ=0TO80:A$=INKEY$:IFA$="" THEN N
EXTJ ELSE COLOR7:GOTO480
460 LOCATE7,20:COLOR14:PRINTM$;:FORJ=0TO80:A$=INKEY$:IFA$=""THENNEXTJ:NEXTI
470 COLOR7
480 I=2:NEXTI:CLS:LOCATE0,3:PRINT"  ｺﾙﾄﾞﾝ ｼﾚｲｶﾝ ﾊ ｱﾅﾀ ﾆ ｶｷﾉﾆﾝﾑ ｦ ｼﾚｲｽﾙ":PRINT
490 PRINT"1) ｵｾﾝﾄﾞ ";:COLOR2:PRINT"100%";:COLOR7:PRINT"ｲﾅｲﾆ ｵｵｸﾉ ｹﾝｷｭｳｲﾝ ｦ ｶｸﾘｽﾙ
｡":PRINT
500 PRINT"2) ﾃｷｶｸﾅ ﾊﾝﾀﾞﾝﾆﾖﾘ ｷｭｳｼﾂｦ ｱｷﾗﾒﾃ ﾎﾞｳｷﾞｮﾍｷ"
510 PRINT"   ｦ ﾌｶｼｮ ｽﾍﾞﾃﾆ ﾌｾﾂｽﾙ｡ ﾎﾞｳｷﾞｮﾍｷ ﾊ ﾐｷﾞｼﾀ"
520 PRINT"   ﾉ ｶｸﾉｳｺ ﾖﾘ ﾊﾝｼｭﾂ ｼｮｳｽﾙ｡":PRINT
530 PRINT"3) ";:COLOR4:PRINT"BETA/]C";:COLOR7:PRINT" ｺﾝﾄﾛｰﾙ ｺﾏﾝﾄﾞﾊ ﾂｷﾞﾉ 4 ｼﾚｲﾃﾞｱ
ﾙ"
540 PRINT"                  :CURSOR UP KEY":SYMBOL(112,98),CHR$(30),2,1,7:SYMBOL(208
,96),CHR$(30),2,1,7
550 PRINT"        |         :CURSOR LEFT KEY":SYMBOL(208,104),CHR$(29),2,1,7
560 PRINT"       -";:COLOR4:PRINT"♣";:COLOR7:PRINT"-       :CURSOR DOWN KEY":SYMBOL
(82,113),CHR$(29),2,1,7:SYMBOL(142,113),CHR$(28),2,1,7:SYMBOL(208,112),CHR$(31),
2,1,7
570 PRINT"        |         :CURSOR RIGHT KEY":SYMBOL(208,120),CHR$(28),2,1,7
580 PRINT"                 ";:COLOR4:PRINT"♣";:COLOR7:PRINT":BETA/]C":COLOR7:SYMBOL(
112,127),CHR$(31),2,1,7
590 PRINT:PRINT
600 PRINT"4) ﾆﾝﾑｶﾞ ｼｭｳﾘｮｳｼﾀﾗ ｽﾊﾞﾔｸ ";:COLOR2:PRINT"Return";:COLOR7:PRINT" ｷｰ ｦｵｼ
ﾃ"
610 PRINT"   ｶﾝﾘｮｳ ﾎｳｺｸ ﾄ ｽﾙ｡"
620 M$=" Hit any key to start "
630 FORJ=0TO1:J=0:LOCATE8,22:COLOR6:PRINTM$;:FORI=0TO80:A$=INKEY$:IF A$="" THEN
NEXTI ELSE COLOR7:GOTO660
640 LOCATE8,22:COLOR14:PRINTM$;:FORI=0TO80:A$=INKEY$:IF A$="" THEN NEXTI:NEXTJ
650 COLOR7
660 J=2:NEXTJ:RANDOMIZE TIME/4+ASC(A$)+RND(2)*95
670 CLS:COLOR7:RETURN
680 IF Z=&HE8 THEN LOCATEXS,YS:COLOR1:PRINT"♠";:RETURN
690 IF Z=&H87 THEN LOCATEXS,YS:COLOR6:PRINT"■";:RETURN
700 RETURN
710 GET@A(25,6)-(30,6),IY:M$="ｶｸﾘ ｼﾂ":LOCATE25,6:PRINTM$:GET@A(25,6)-(30,6),IX:F
ORI=0TO10:PUT@A(25,6)-(30,6),IX:FORJ=0TO200:NEXT:PUT@A(25,6)-(30,6),IY:FORJ=0TO2
00:NEXT:NEXT
720 M$="ﾎﾞｰｷﾞｮﾍｷ":LOCATE24,11:PRINTM$
730 FORK=0TO10
740 IX(0)=&H687:FORI=0TO6:PUT@A(PX(I),PY(I))-(PX(I),PY(I)),IX:NEXT
750 FORJ=0TO200:NEXT
760 IX(0)=&H420:FORI=0TO6:PUT@A(PX(I),PY(I))-(PX(I),PY(I)),IX:NEXT
770 FORJ=0TO200:NEXT
780 NEXT
790 COLOR5:LOCATE24,11:PRINTSPC(8);:RETURN
800 LINE(16,32)-(335,175),PSET,0,BF
810 HI=0:FORY=5TO8:FORX=22TO33:C=SCREEN(X,Y,0):IF C=&HE8 THEN HI=HI+1
820 NEXT:NEXT
830 KJ=0:FORI=0TO6:C=SCREEN(PX(I),PY(I),0):IF C=&H87 THEN KJ=KJ+1
840 NEXT
850 LOCATE 3,5:PRINT" ｵｾﾝﾄﾞ    ";STR$(LK/2+OD);"%"
860 LOCATE 3,6:PRINT" ｶｸﾘｼﾀ ﾋﾄ";STR$(HI);" ﾋﾄ"
870 LOCATE 3,7:PRINT" ﾎﾞｳｷﾞｮﾍｷ";STR$(KJ);" ﾌﾞﾛｯｸ":GOSUB950
880 M$=" Try again ? "
890 FORI=0 TO 1 STEP 1:I=0:LOCATE 2,20:COLOR 6:PRINT M$;:FOR J=0 TO 120:A$=INKEY
$:IF A$="" THEN NEXTJ ELSE 910
900 LOCATE 2,20:COLOR14:PRINT M$;:FOR J=0 TO 120:A$=INKEY$:IF A$="" THEN NEXTJ:N
EXTI
910 I=2:NEXTI:IF A$="Y" OR A$="y" THEN COLOR7,0:RUN 20
920 IF A$<>"N"ANDA$<>"n" THEN 890
930 COLOR7,0:CLS:LOCATE10,10:PRINT"NICE TO MEET YOU":LOCATE10,12:PRINT"SEE YOU A
GAIN":LOCATE0,20:END
940 BEEP1:LOCATET,U:COLOR2:PRINT"●";:BEEP0:GOTO40
950 U=INT(HI*KJ/(LK/2+OD+1)*100)
960 IF (KJ=7)*(HI<=5)THEN LOCATE 3,CSRLIN+2:PRINT"ｾｲｾｷﾋｮｳｶ ";STR$(U);" POINT":PR
```

```
INT:GOTO1010
970 IF (KJ=7)*(HI>5)THEN LOCATE 3,CSRLIN+2:PRINT"ｾｲｾｷﾋｮｳｶ ";STR$(U);" POINT":PRI
NT:GOTO1040
980 LOCATE 2,CSRLIN+2:PRINT"ｻﾞﾝﾈﾝ! ﾎﾞｳｷｮﾍｷ":LOCATE 2,CSRLIN:PRINT"ｶﾞ ﾌｶﾝｾﾞﾝﾃﾞ ﾁｷ
ｭｳ ﾊ"
990 LOCATE 2,CSRLIN:COLOR5:PRINT"ALDEBARAN ";:COLOR7:PRINT"ﾆ ｵｾﾝ":LOCATE2,CSRLIN
:PRINT"ｻﾚﾃｼﾏｲﾏｼﾀ｡":RETURN
1000 END
1010 LOCATE2,CSRLIN+2:PRINT"ｵﾒﾃﾞﾄｳ ";:COLOR5:PRINT"ALDEBARAN";:COLOR7:LOCATE2,CS
RLIN+1:PRINT"ﾉ ｼﾝﾆｭｳ ﾊ ﾌｾｶﾞﾚﾏｼﾀ｡"
1020 LOCATE2,CSRLIN:PRINT"ﾃﾞﾓｷﾞｾｲｼｬ ﾉ ｶｽﾞｶﾞ":LOCATE2,CSRLIN:PRINTSTR$(10-HI);" ﾆ
ﾝ ﾃﾞｵｵｽｷﾞﾏｽ｡"
1030 RETURN
1040 LOCATE2,CSRLIN+2:PRINT"ｵﾒﾃﾞﾄｳ ";:COLOR5:PRINT"ALDEBARAN";:COLOR7:LOCATE2,CS
RLIN+1:PRINT"ﾉ ｼﾝﾆｭｳ ﾊ ﾌｾｶﾞﾚﾏｼﾀ｡"
1050 IF HI=10 THEN 1060 ELSE LOCATE2,CSRLIN:PRINT"ｱﾅﾀﾉﾁｶﾗﾃﾞ ｷﾞｾｲｼｬﾓ":LOCATE2,CSR
LIN:PRINTSTR$(10-HI);"ﾆﾝ ﾃﾞｽﾐﾏｼﾀ｡":GOTO1070
1060 LOCATE2,CSRLIN:PRINT"ｱﾅﾀﾉ ﾁｶﾗﾃﾞ ｷﾞｾｲｼｬﾓ":LOCATE2,CSRLIN:PRINT"ﾃﾞﾏｾﾝ ﾃﾞｼﾀ｡"
1070 LOCATE2,CSRLIN+1:PRINT"ｺﾞｸﾛｳｻﾏﾃﾞｼﾀ｡":RETURN
1080 CLS:LOCATE0,0
1090 PRINT"┌┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┬┐"
1100 PRINT"|                                  | ｵｾﾝﾄﾞ    0   % |"
1110 PRINT"└┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┴┘"
1120 X=35:Y=15:LOCATEX,Y:COLOR4:PRINT"♣":COLOR7:L=1:M=1:YK=-1
1130 '
1140 RESTORE:FORI=0TO6:READ PX(I),PY(I):NEXT
1150 LOCATE0,3:PRINT"XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
1160 PRINT"X";:COLOR5:PRINT"                  ||  ";:COLOR7:PRINT"XXXXXXXXXXXXXXXX"
1170 PRINT"X";:COLOR5:PRINT"                  |└─";:COLOR7:PRINT"X           X"
1180 PRINT"X";:COLOR5:PRINT"          ┌─┐   |   ";:COLOR7:PRINT"
1190 PRINT"X";:COLOR5:PRINT"                     ";:COLOR7:PRINT"X          ─┘"
1200 PRINT"X";:COLOR5:PRINT"            | |  ┌┐ ┌";:COLOR7:PRINT"X─────────"
1210 PRINT"X";:COLOR5:PRINT"                     ";:COLOR7:PRINT"XXXXX XXXXXXXXXX"
1220 PRINT"X";:COLOR5:PRINT"───┐                ":COLOR7
1230 PRINT"X";:COLOR5:PRINT"   |   ";:COLOR2:PRINT"█████";:COLOR5:PRINT"
  ";COLOR7
1240 PRINT"X";:COLOR5:PRINT"   |   ";:COLOR2:PRINT"█┌";:COLOR5:PRINT"XRB";:COLOR
2:PRINT"┐";:COLOR10:PRINT"|";:COLOR5:PRINT" ┌───";:COLOR7:PRINT"XXXXXXXXXXXXXX"
1250 PRINT"X";:COLOR5:PRINT"   ┘     ";:COLOR2:PRINT"█";:COLOR5:PRINT"ALD-7";:COLO
R10:PRINT"|";:COLOR5:PRINT"  └──";:COLOR7:PRINT"X"
1260 PRINT"X";:COLOR5:PRINT"        ";:COLOR10:PRINT"______";:COLOR5:PRINT"
 ";:COLOR7:PRINT"X   ";:COLOR6:PRINT"████";:COLOR7:PRINT"    X"
1270 PRINT"X";:COLOR5:PRINT"──┐                 ";:COLOR7:PRINT"X        X"
1280 PRINT"X";:COLOR5:PRINT"  |             |   ";:COLOR7:PRINT"X  ";:COLOR6:PRI
NT"████";:COLOR7:PRINT"    X"
1290 PRINT"X";:COLOR5:PRINT"  |     ┌┬─────┬┬┘ ";:COLOR7:PRINT"X    X"
1300 PRINT"X";:COLOR5:PRINT"        ┘      └┘  ";:COLOR7:PRINT"X  ";:COLOR6:PRI
NT"████";:COLOR7:PRINT"    X"
1310 PRINT"X";:COLOR5:PRINT"  ┘                 ";:COLOR7:PRINT"X        X"
1320 PRINT"X";:COLOR5:PRINT"          ┌┐   ┌┐  ";:COLOR7:PRINT"X  ";:COLOR6:PRI
NT"████";:COLOR7:PRINT"    X"
1330 PRINT"X";:COLOR5:PRINT"          ||   ||  ";:COLOR7:PRINT"X        X"
1340 PRINT"XXXXXXX  XXXXXXXX XXXX  XXXXXXXXX":GOSUB710:I=0
1350 'COLOR 1:LOCATE2,12:PRINT"♠":LOCATE3,7:PRINT"♠":LOCATE13,5:PRINT"♠":LOCATE1
2,19:PRINT"♠":LOCATE9,16:PRINT"♠":LOCATE5,18:PRINT"♠":LOCATE5,5:PRINT"♠":LOCATE5
,15:PRINT"♠":LOCATE18,19:PRINT"♠":LOCATE7,19:PRINT"♠":BEEP1:BEEP0:BEEP1:BEEP0:RE
TURN
1360 XM=INT(RND(1)*13)+2:YM=INT(RND(1)*18)+3:A=40*YM+XM+&H400
1370 IF(SCREEN(XM,YM,0)=32) AND (SCREEN(XM,YM-1,0)=32) AND (SCREEN(XM,YM+1,0)=32
) AND (SCREEN(XM+1,YM,0)=32) AND (SCREEN(XM-1,YM,0)=32) THEN 1390 ELSE 1360
1380 '
1390 HT=98
1400 COLOR1:LOCATEXM,YM:PRINT"♠";:I=I+1:IF I<10 THEN 1360
1410 COLOR7:FORI=0TO5:BEEP1:FORJ=0TO100:NEXT:BEEP0:NEXT:RETURN
1420 DATA 21,6,25,9,21,10,21,11,7,22,8,22,17,22
1430 CLS:LOCATE11,8:COLOR5:SYMBOL(100,10),"ALDEBARAN",6,4,4
1440 LOCATE17,8:PRINT"Prat 1"
1450 LOCATE10,17:COLOR7:PRINT"HUDSON SOFT SAPPORO"
1460 LOCATE10,20:PRINT"COPYRIGHT (C)  1981"
1470 FORI=0TO6000:NEXT:RETURN
```

くやしさいっぱいのFM-8ユーザーのための

製作実験記事

TTL2個でつくる倍速基板

★ご注意★

ハードを改造し，万一，異常が起きた場合，メーカーのメンテナンスが受けられなくなることがあります。ハードの基礎知識およびテクニックに自信のない方は，絶対に改造しないでください。

FM-7の発売で，くやしい思いをしたFM-8ユーザーも多いことでしょう。今回，TTLたったの2個で倍速改造基板を作ることに成功しましたので，ご報告いたします。愛機FM-8の機能アップにお役立てください。

米村英明

はじめに

FM-8のMAIN CPUは約1.2MHz，SUB CPUは約1MHzで作動しています。それらを2MHzに切り換え，FM-8をFM-7並みにスピードアップさせる回路を〝TTLたった2個〟で作ってしまおうというわけです。しかし，部品数が少ないだけにFM-8内の回路に負うところが多く，パターンカットこそありませんが，引き出し線が多く，配線ミスなどがあるとFM-8の故障に直結しますので，初心者の方はご遠慮ください。また同じFM-8でも，物によっては2MHzの動作に付いてこれないものもありますので，注意してください。

回　路

FM-8を高速化するには，単にクロックを速くすればよいではないか，と考える方もいると思いますが，1.2MHzのクロックはタイマ，ビープ音，RS-232Cなど，各種I/Oに使用しており，また，CPUのクロックを2MHzにしたままですと，テープからのLOAD，SAVEができなくなります。

したがって，CPUのクロックは使用状態に合わせて切り換えてやる必要があるわけです。だからといって，無作為にクロックを切り換えるとFM-8は暴走します。要するにこの回路で使用する2個のTTLは，なんのことはない，クロック切り換えのためのタイミングをとるものなのです。74LS74の中の2個のD-FFで2種のクロックのタイミングをとり，74LS157のデータセレクタでクロックを切り換えているだけで，詳しい動作原理は，**図1**の回路図を見ればすぐに理解できると思います。

部品表

MC68B09（できるだけモトローラ製が良い）……………………2個
74LS74 ……………………1個
74LS157 ……………………1個
プリント基板……………………1枚
テスタ用クリップ……………………9個
配線用の線材……………………少々

製　作

まず，最初にクロック切り換えの基板を作ります。部品はTTL2個だけなので簡単ですが，くれぐれも誤配線や短絡などをしないよう気をつけてください。なお，FM-8本体基板上に配線する部分は，端子を作っておくようにして（**写真**参照），FM-8の基板と配線しながら組み立てるようなことはしないでください。クロック切り換えの基板ができあがったら，誤配線がないかチェックします。とくに電源間のショートは致命的な故障の原因となりますので，注意してください。

次に**図2**のように，2個の68B09の38番ピンを曲げて，FM-8に初めから付いている6809と差し換えてください。そして，クロック切り換え基板を，**図3**のとおりにFM-8に接続します。図中のM14などの番号は，FM-8の基板上に付けられているTTLなどの番号です。各TTL，LSIのピンに引き出し線をつなぐときは，必ずテスタ用のクリップなどを使用し，また，隣のピンとの短絡にも注意してください。

テスト

配線ミスなどのチェックが完了すれば，いよいよFM-8の電源SWをONにします。そのとき，何も表示されない場合は，すぐ

図2

図4

に電源を切って回路をチェックしてください。正常に動かないときは，残念ですが，修理に出すほかありません。電源を入れたときに暴走する場合は，リセット時に正しく1.2MHzのクロックがセレクトされていず，ブートROMが追いついてこれなくなるのが原因です。クロック関係の配線をもう一度チェックしましょう。しかし，中には2MHzでも追いついてくれるブートROMを持ったFM-8もあります。

正常にリセット時のメッセージは出るのだが，キャラクタがおかしい場合もあります。これはキャラクタジェネレータが2MHzについてこれないためで，M56のROMを**図4**のようにします。

正常に動くようでしたら，次のプログラムを実行してみてください。

```
10  TIME$="00:00:00"
20  FOR  I=1  TO  10000
30  NEXT
40  PRINT  TIME$
```

1.2MHzの場合は，10秒かかります。次に

POKE　&HFD00, &HC4

を実行した後，前のプログラムを実行して

図3

M14
74LS273
6番ピン
38 37
MC68B09
MAIN CPU
TP1
MAIN CPU φ
SFD00
RESET
1.2MHz
2MHz
Vcc
GND
38
1 7
MC68B09
SUB CPU
14番ピン
M186
74LS163
漢　字 ROM
CN9

みてください。5秒程で実行が終われば，正常です。それ以外の結果になった方は，また配線をチェックしてください。なお，前述の異常は数時間たった後に現れる場合がありますが，そういった経時変化の大きいＦＭ-8を買った方は，運が悪かったと思って，長時間の高速化はあきらめてください。

使い方

この基板は，＄ＦＤ00番地のb_3をセットすると，クロックが切り換わるようになっていますが，カセットを使用するときは，自動的に1.2MHzに戻るようになっています。

LOAD，SAVE，MOTORなどの命令や，エラーを出したときは，1.2MHzに戻りますから，その後，倍速で使いたいときは，もう一度，POKE　&HFD00，&HC4を実行してください。

おまけ

最後に，ちょっとしたBASICのゲームを付けておきます。通常ではちょっと遅すぎますが，倍速にするとちょうどよくなります。

ひまつぶしに作ったゲームなので，それほどのものではありませんが，どうぞお楽しみください。

リスト　倍速FM-8用プログラムリスト（FM-7,11でもOK）

（FM-11の方は最初に SCREEN 4 を実行してください）

```
1 DEFINT A-Z
2 DIM B(50),C(1000),M(5,10),A(3),E(10),T(50)
3 DEF FNX(MN,DN)=M(0,MN)+M(2,MN)*DN
4 DEF FNY(DY,DN)=DY*DN
5 WIDTH80,25
6 RANDOMIZE TIME
7 COLOR4:LOCATE28,10:PRINT"<<< 10秒 ﾎﾄﾞ ｵﾏﾁ ｸﾀﾞｻｲ >>>"
8 GOSUB45
9 SC=0:CN=8:TX=320:TY=80
10 GOSUB15:RD=1
11 GOSUB32:GOSUB51
12 NN=8-2*((RD-1) MOD 3):GOSUB54
13 IF RD MOD 3=0 THEN CN=CN/2:IF CN<1 THEN CN=1
14 GOSUB77:RD=RD+1:GOTO11
15 X0=64:Y0=64
```

```
16 FORI=3TO0STEP-1:SYMBOL(X0+I,Y0+I),"MISSILE COMMAND",4,2,4:NEXTI
17 SYMBOL(X0,Y0),"MISSILE COMMAND",4,2,6
18 SYMBOL(289,97),"KEY",2,1,1
19 SYMBOL(288,96),"KEY",2,1,5
20 SYMBOL(296,112),"8",2,1,3
21 SYMBOL(305,113),CHR$(30),2,1,1
22 SYMBOL(304,112),CHR$(30),2,1,5
23 SYMBOL(264,120),"4 2 6",2,1,3
24 SYMBOL(273,121),CHR$(29)+" "+CHR$(31)+" "+CHR$(28)+" ...MOVE",2,1,1
25 SYMBOL(272,120),CHR$(29)+" "+CHR$(31)+" "+CHR$(28)+" ...MOVE",2,1,5
26 SYMBOL(273,137),"A S D ...FIRE",2,1,1
27 SYMBOL(272,136),"A S D ...FIRE",2,1,5
28 SYMBOL(49,169),"PUSH [RETURN] KEY",4,1,2
29 SYMBOL(48,168),"PUSH [RETURN] KEY",4,1,6
30 A$=INPUT$(1):IF ASC(A$)=13 THEN RETURN ELSE 30
31 LOCATE0,24:FORI=1TO25:PRINT:FORJ=1TO30:NEXTJ,I:RETURN
32 COLORRD MOD4+4:IFRD=1THEN38
33 BN=0:FORI=0TO639:IFPOINT(I,167)<>0THENBN=BN+RD-1:GOSUB93
34 NEXTI
35 FORI=1TO3:BN=BN+A(I)*100:GOSUB93:NEXT I:SC=SC+BN
36 SYMBOL(66,129),"BONUS:"+STR$(BN),4,1,6
37 SYMBOL(64,128),"BONUS:"+STR$(BN),4,1,4
38 GOSUB31
39 SYMBOL(193,177),"ROUND"+STR$(RD),4,1,4
40 SYMBOL(192,176),"ROUND"+STR$(RD),4,1,6
41 SYMBOL(129,193),"･="+STR$(RD*10)+" POINT",4,1,3
42 SYMBOL(128,192),"･="+STR$(RD*10)+" POINT",4,1,7
43 GOSUB31
44 RETURN
45 RESTORE96:FORI=0TO50:READB(I):NEXTI
46 RESTORE101:FORI=0TO999:READC(I):NEXTI
47 CLS:LINE(0,168)-(639,183),PSET,1,BF
48 PUT@A(0,160)-(639,167),C,PSET
49 FORI=0TO2:PUT@A(I*304,160)-(I*304+31,167),B,PSET:NEXTI
50 RETURN
51 LINE(0,168)-(639,183),PSET,1,BF
52 PUT@A(0,160)-(639,167),C,PSET
53 COLOR7:FORI=0TO2:PUT@A(I*304,160)-(I*304+31,167),B,PSET:A(I+1)=10:LOCATEI*38,
21:PRINTA(I+1);:NEXTI:RETURN
54 DY=160/NN:FORN=4TO6:GET@A(TX-8,TY-4)-(TX+8,TY+4),T,G:GOSUB55:FORDN=1TONN:COLO
R6:LOCATE5,0:PRINT"SCORE:";SC:FORMN=1TOCN:GOSUB61:GOSUB58:NEXTMN,DN:GOSUB72:NEXT
N:RETURN
55 LOCATE5,0:PRINT"NOW SEARCHING TARGET...":FORI=1TO10:M(0,I)=INT(RND*640):M(4,I
)=M(0,I):M(5,I)=0
56 J=RND*640:IFPOINT(J,167)=0THEN56
57 M(1,I)=J:M(2,I)=(M(1,I)-M(0,I))/NN:M(3,I)=1:NEXTI:LOCATE5,0:PRINT"
                                      ":RETURN
58 IFM(3,MN)=0THENRETURN
59 M(4,MN)=FNX(MN,DN):M(5,MN)=FNY(DY,DN):LINE(FNX(MN,DN-1),FNY(DY,DN-1))-(M(4,MN
),M(5,MN)),PSET,N:IFPOINT(M(4,MN),M(5,MN)+1)=-1THENGOSUB80
60 RETURN
61 FORI=0TO2:LOCATEI*38,21:PRINTA(I+1);:NEXTI:A$=INKEY$:PX=TX:PY=TY
62 IFA$="8"THENIFTY>8THENTY=TY-8
63 IFA$="2"THENIFTY<120THENTY=TY+8
64 IFA$="6"THENIFTX<624THENTX=TX+16
65 IFA$="4"THENIFTX>16THENTX=TX-16
66 PUT@A(PX-8,PY-4)-(PX+8,PY+4),T,PSET:GET@A(TX-8,TY-4)-(TX+8,TY+4),T,G:SYMBOL(T
X-8,TY-4),"+",2,1,5
67 IFA$="A"THENA=1:GOTO71
68 IFA$="S"THENA=2:GOTO71
69 IFA$="D"THENA=3:GOTO71
70 RETURN
71 GOSUB78:RETURN
72 FORC=7TO0STEP-7:FORR=1TO32:FORMN=1TOCN:IFM(3,MN)=0THEN74
73 CIRCLE(M(1,MN),168),R,C,,.5,1
74 NEXTMN,R,C:GOSUB86
75 FORI=0TO2:IFPOINT(I*304+16,167)=0THENA(I+1)=0
76 NEXTI:LINE(0,0)-(639,159),PRESET,,BF:RETURN
77 GET@A(0,160)-(639,167),C,G:RETURN
78 IFA(A)=0THENRETURN
79 A(A)=A(A)-1:LINE((A-1)*304+16,166)-(TX,TY),PSET,3:FORR=1TO32:CIRCLE(TX,TY),R,
```

```
7:NEXTR:GET@A(TX-8,TY-4)-(TX+8,TY+4),T,G:GOSUB94:LINE((A-1)*304+16,166)-(TX,TY),
PRESET:RETURN
80 IF DN=NN OR ((M(4,MN)<TX+8 AND M(4,MN)>TX-8) AND (M(5,MN)<TY+4 AND M(5,MN)>TY
-4)) THENRETURN
81 M(3,MN)=0:SC=SC+RD*10
82 LINE(M(0,MN),0)-(M(4,MN),M(5,MN)),PRESET:FORR=4TO32STEP4:CIRCLE(M(4,MN),M(5,M
N)),R,7:NEXTR:GOSUB94:RETURN
83 IF DN=NN OR ((M(4,MN)<TX+8 AND M(4,MN)>TX-8) AND (M(5,MN)<TY+4 AND M(5,MN)>TY
-4)) THENRETURN
84 IFM(3,MH)=1THENLINE(M(0,MH),0)-(M(4,MH),M(5,MH)),PRESET:FORR=4TO32STEP4:CIRCL
E(M(4,MH),M(5,MH)),R,7:NEXTR:M(3,MH)=0:MH=0:SC=SC+RD*10
85 RETURN
86 FORI=0TO639:IFPOINT(I,167)<>0THEN RETURN
87 NEXT I
88 SYMBOL(50,82),"<<< GAME OVER >>>",4,2,2
89 SYMBOL(48,80),"<<< GAME OVER >>>",4,2,6
90 SYMBOL(66,129),"SCORE:"+STR$(SC),4,1,4
91 SYMBOL(64,128),"SCORE:"+STR$(SC),4,1,6
92 RETURN8
93 LOCATE5,3:PRINT"BONUS:";BN:RETURN
94 MH=1:WHILEMH<CN:IFPOINT(M(4,MH),M(5,MH)+1)<>0THENGOSUB83
95 MH=MH+1:WEND:RETURN
96  DATA  0 ,-32768 , 127 , 32512 , 896 , 224 , 3196 , 28696 , 14078 , 31222
97  DATA  10239 ,-2 , 20479 ,-3 ,-12289 ,-3 , 0 , 0 , 0 , 0
98  DATA  0 , 0 , 0 , 0 , 0 , 0 , 225 , 6144 , 251 ,-1152
99  DATA  255 ,-128 , 0 ,-32768 , 127 , 32512 , 896 , 224 , 3072 , 24
100  DATA  12802 , 6182 , 9199 ,-24734 , 17407 ,-31 ,-15361 ,-31 , 0 , 0
101  DATA  0 ,-32768 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
102  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
103  DATA -32768 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
104  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,-32768
105  DATA  127 , 32512 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
106  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 127
107  DATA  32512 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
108  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 127 , 32512
109  DATA  896 , 224 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
110  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 896
111  DATA  224 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
112  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 896 , 224
113  DATA  3196 , 28696 , 0 , 17404 , 240 , 247 ,-256 , 960 , 511 ,-32528
114  DATA  16128 , 29167 , 8160 , 4080 , 3840 , 32761 ,-7173 ,-14464 , 16128 , 3
196
115  DATA  28696 , 0 , 28672 , 32701 ,-8191 ,-8192 , 961 ,-8065 ,-16384 , 0
116  DATA  30960 , 511 ,-32768 , 0 , 25057 ,-8192 , 15 , 7936 , 3196 , 28696
117  DATA  14078 , 31222 , 0 , 32759 , 32767 ,-24713 ,-256 , 18375 ,-7681 , 2431
3
118  DATA -256 , 30719 ,-129 , 32767 ,-4352 , 32639 , 32755 ,-193 ,-2304 , 14078

119  DATA  31222 , 0 , 30975 ,-65 ,-7 ,-7936 , 32767 ,-7169 ,-12448 ,-16640
120  DATA  32767 ,-32257 ,-76 , 3840 , 32761 ,-7041 ,-24705 , 32512 , 14078 , 31
222
121  DATA  10239 ,-2 , 0 , 30719 ,-129 , 32767 ,-256 , 30591 ,-1 ,-129
122  DATA  32512 , 32767 ,-2049 , 32639 ,-256 , 32767 ,-2049 , 32759 ,-256 , 102
39
123  DATA -2 , 0 , 32759 , 32639 ,-2049 , 30464 , 32639 ,-1 ,-9 ,-2304
124  DATA  30719 ,-6281 ,-2177 ,-256 , 32767 ,-137 , 32759 ,-2304 , 10239 ,-2
125  DATA  20479 ,-3 , 0 , 32639 ,-2049 ,-2049 , 32512 , 32767 , 30583 ,-2049
126  DATA -2304 , 32631 ,-1 ,-9 ,-2304 , 32767 ,-129 ,-1 ,-256 , 20479
127  DATA -3 , 0 , 30719 ,-2049 , 32631 ,-256 , 30711 ,-2185 , 30719 , 32512
128  DATA  32631 ,-1 ,-1 ,-2304 , 30583 ,-1 ,-1 , 32512 , 20479 ,-3
129  DATA -12289 ,-3 , 0 , 32767 ,-9 ,-1 ,-2304 , 32767 ,-1 ,-9
130  DATA -256 , 32767 , 32759 ,-2049 , 32512 , 30711 ,-1 ,-2049 , 32512 , 20479

131  DATA -3 , 0 , 32639 ,-9 ,-1 ,-2304 , 32767 , 32767 ,-1 ,-256
132  DATA  32767 , 32767 , 32767 ,-256 , 32767 , 32767 ,-2049 ,-256 , 20479 ,-3
133  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
134  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
135  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
136  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
137  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
138  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
```

```
139  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
140  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
141  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
142  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
143  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
144  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
145  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
146  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
147  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
148  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
149  DATA  0 , 0 , 0 , 8 ,-32768 , 136 , 0 , 8 , 0 ,-32768
150  DATA  0 , 2048 , 128 ,-32768 , 0 , 128 ,-32760 , 128 , 2048 , 0
151  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 128 , 0
152  DATA  0 , 0 , 8 ,-32768 , 0 , 2048 , 128 , 0 , 0 , 0
153  DATA  225 , 6144 , 0 , 2048 , 128 ,-32768 , 0 , 2176 , 0 , 128
154  DATA -32768 , 0 , 2048 , 128 , 0 , 0 , 2048 ,-32760 , 0 , 225
155  DATA  6144 , 0 , 8 ,-32640 , 2048 ,-32768 , 128 , 0 , 8 , 2048
156  DATA  2048 , 2184 , 2176 , 0 , 0 , 136 ,-32760 , 2048 , 225 , 6144
157  DATA  251 ,-1152 , 0 , 128 , 2048 , 2048 ,-32768 , 0 ,-30584 , 2048
158  DATA  2048 , 136 , 0 , 8 , 2048 , 0 , 128 , 0 , 0 , 251
159  DATA -1152 , 0 , 2048 , 2048 ,-32632 , 0 , 2056 , 2184 ,-30720 ,-32768
160  DATA  136 , 0 , 0 , 2048 , 2184 , 0 , 0 ,-32768 , 251 ,-1152
161  DATA  255 ,-128 , 0 , 0 , 8 , 0 , 2048 , 0 , 0 , 8
162  DATA  0 , 0 ,-32760 , 2048 ,-32768 , 2056 , 0 , 2048 ,-32768 , 255
163  DATA -128 , 0 , 128 , 8 , 0 , 2048 , 0 ,-32768 , 0 , 0
164  DATA  0 ,-32768 ,-32768 , 0 , 0 ,-32768 , 2048 , 0 , 255 ,-128
165  DATA  0 ,-32768 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
166  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
167  DATA -32768 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
168  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,-32768
169  DATA  127 , 32512 , 0 , 0 , 0 , 0 , 0 ; 0 , 0 , 0
170  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 127
171  DATA  32512 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
172  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 127 , 32512
173  DATA  896 , 224 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
174  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 896
175  DATA  224 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
176  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 896 , 224
177  DATA  3072 , 24 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
178  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 3072
179  DATA  24 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
180  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 3072 , 24
181  DATA  12802 , 6182 , 0 , 3002 ,-31728 , 2696 , 256 , 1165 ,-32702 ,-12184
182  DATA  0 , 19208 , 17539 ,-8192 ,-30720 , 16771 ,-23512 , 3513 , 19200 , 128
02
183  DATA  6182 , 0 , 2096 , 9347 , 4112 , 0 , 545 , 16906 , 2208 ,-32768
184  DATA  1026 , 0 ,-31736 ,-32768 , 2184 , 19520 , 130 , 2048 , 12802 , 6182
185  DATA  9199 ,-24734 , 0 , 3006 ,-18792 ,-9224 ,-19712 , 3725 ,-1821 ,-11524
186  DATA -5120 , 20316 , 20239 , 28628 ,-29440 , 26499 ,-4996 ,-24709 , 28416 ,
 9199
187  DATA -24734 , 0 , 18618 ,-23061 ,-10222 ,-16384 , 32427 ,-13634 , 22698 ,-1
3056
188  DATA  23847 ,-21620 ,-17014 ,-32512 , 11721 ,-14358 ,-31685 ,-14080 , 9199
,-24734
189  DATA  17407 ,-31 , 0 , 14333 ,-453 ,-3 ,-2560 , 24379 ,-513 ,-20483
190  DATA -768 , 7935 ,-24642 ,-8195 , 32512 , 32623 ,-9217 ,-266 ,-256 , 17407
191  DATA -31 , 0 , 32743 ,-9217 ,-1809 ,-2560 , 32223 ,-16899 ,-41 ,-4352
192  DATA  31743 ,-115 , 31679 ,-256 , 32735 ,-16385 ,-8201 ,-10496 , 17407 ,-31

193  DATA -15361 ,-31 , 0 , 32767 ,-1 ,-1 ,-256 , 32767 ,-1 ,-1
194  DATA -256 , 32767 ,-1 ,-1 ,-256 , 32767 ,-1 ,-1 ,-256 , 17407
195  DATA -31 , 0 , 32767 ,-1 ,-1 ,-256 , 32767 ,-1 ,-1 ,-256
196  DATA  32767 ,-1 ,-1 ,-256 , 32767 ,-1 ,-1 ,-256 , 17407 ,-31
197  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
198  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
199  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
200  DATA  0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
```

初心者のための
F-BASICプログラミングアプローチ

山田 陽一郎

1 プロローグ

FM-8が発売されてから1年余り過ぎた今，普及タイプのFM-7がリリースされ，FMユーザーの輪もいっそう広がってきた。だが，せっかく入手しても種々の事情により十分使いこなしていない方も多いのではないだろうか。

本稿では，F-BASICに少しでも慣れていただくために，主要コマンドのいくつかを取り上げ解説し，必要に応じてサブルーチン形式のプログラムを紹介する。初心者ユーザーの助力になれば幸いである。

2 入力命令

2-1 LINE INPUT

プログラムを作成するうえで，入力命令は特に重要である。基本形としてINPUTがあるが，LINE INPUTは拡張されたINPUTと考えることができる。例を次に示す。

```
10 INPUT A$
20 PRINT A$
30 END
RUN
?Yes,I do.
Yes
Ready
```

```
10 LINE INPUT A$
20 PRINT A$
30 END

RUN
?Yes,I do.
Yes,I do.
Ready
```

このように，INPUTではカンマを区切りとして分割されてしまうが，LINE INPUTでは，RETURNが入力されるまでを文字変数として代入する。すなわち，INPUTで生じた不都合がなくなるが，場合によっては不要の値まで取り込む恐れがあるので注意が必要である。

一般に，数値データを複数個入力するとき，仮にデータの個数がわかっている場合，次の2とおりの書き方ができる。

```
(a)
 10 DIM A(6)
 20 INPUT A(0),A(1),A(2),
  A(3),A(4),A(5),A(6)


(b)
 10 DIM A(100)
 20 FOR I=0 TO 100
 30   INPUT A(I)
 40 NEXT I
```

（a）は単純にINPUTの後に続ける方法で，ユーザーはデータをカンマで区切って入力する。ただし，この方法だと入力データが数十個くらいになってくると，入力ルーチンをその分だけ並べる必要があり，手間を要する。このような場合に有効な方法が，（b）のFOR～NEXTループを用いる手法である。（b）のようにすれば，入力数はループの終値いかんで汎用性に富む。しかし，毎回Returnキーを入力しなければならず，また誤入力の訂正もできないという短所がでてくる。

そこで，これらの点を解消すべきプログラムを次に示す。入力には，LINE INPUT文を用い，入力個数があらかじめ決まっていないときにもホスト側が教えてくれる長所を持つ。

```
10 DIM A(100)
20 LINEINPUT A$
30 N=0
40 J=INSTR(A$,",")
50 IF J<>0 THEN A(N)=VAL(
LEFT$(A$,J-1)):N=N+1:A$=R
IGHT$(A$,LEN(A$)-J):GOTO
40
60 IF A$<>"" THEN A(N)=VA
L(A$) ELSE N=N-1
70 PRINT "Number of Data=
";N+1
80 FOR I=0 TO N:PRINT A(I
);NEXT
90 END
```

2-2 INKEY$，INPUT$

INKEY$……キーバッファから1文字を取り出す文字関数

INPUT$……キーボードから指定した文字数だけキーの内容を読み込む

両関数とも，キーボードを押したときのキーの値を持ち返る関数であり，A$=INKEY$，A$=INPUT$(n)のように用いる。

INKEY$は，このプログラム実行時にキーを押していない場合，空文字が与えられる。ほんの一瞬であるため，次のようにキー入力を待つ方法が用いられる。

```
10 A$=INKEY$:IF A$="" THE
N 10
```

これと同様な働きは，INPUT$を用いて次のように表現できる。

```
10 A$=INPUT$(1)
```

3 STRING処理

STRING関数はたくさんあり，各々おもしろい働きをする。反射神経型ゲームなどには用いられないが，思考型ゲームおよびワードプロセッサなど単語入力を要する処理には不可欠な関数である。**表1**にSTRING関数の

表1　文字データ操作命令表

機　　能	予約語	書　　　式
指定コード1文字を与える	CHR$	CHR$(コード) 数式
先頭1文字のコードを与える	ASC	ASC(文字列) 文字式
文字列に変換	STR$	STR$(引数) 数式
数値に変換	VAL	VAL(文字列) 文字式
文字列の取り出し	MID$	MID$(文字列,位置〔,文字数〕) 文字式 数式 数式
文字列の書き換え	MID$	MID$(文字変数名,位置〔,文字数〕)=文字列 数式 数式 文字式
左からの文字列取り出し	LEFT$	LEFT$(文字列,文字数) 文字式 数式
右からの文字列取り出し	RIGHT$	RIGHT$(文字列,文字数) 文字式 数式
特定文字を指定数与える	STRING$	STRING$(文字数,{コード / 数式 / 文字式}) 数式
文字列の長さを与える	LEN	LEN(文字列) 文字式
文字列の存在位置を与える	INSTR	INSTR(〔先頭位置,〕文字列1,文字列2) 数式 文字式 文字式
16進表記の文字列を与える	HEX$	HEX$(引数) 数式
N個の文字列	SPACE$	SPACE$(引数)

種類を示す。

3-1　INSTR

文字列探索のコマンドがINSTRである。押したキーによりプログラムの流れが変わる場合に用いる。次に例として，カーソルキーによりCRT上の文字を動かすプログラムを示す。カーソルキーのように画面に直接表示できないものも，INSTRの引数として使える。

```
10 DIM B(4),C(4)
20 FOR I=1 TO 4:READ B(I)
,C(I):NEXT
30 FOR I=1 TO 4:H$=H$+CHR
$(27+I):NEXT
40 DATA 1,0,-1,0,0,-1,0,1
50 LOCATE 20,10:PRINT "*"
;:X=20:Y=10
60 G$=INPUT$(1):G=INSTR(H
$,G$):IF G=0 THEN 60
70 PRINT " ":X=X+B(G):Y=Y
+C(G):LOCATE X,Y:PRINT "*
":GOTO 60
```

3-2　LEN, MID$, RIGHT$, LEFT$

これらは文字列，とくに単語などを扱うときに用いられる。文字や数値をいくつか並べて表示しようとすると，プログラム中に画面制御命令をいくつか使って位置ぎめを行う必要がある。このような場合，文字変数を用いて処理は文字処理により行うようにすれば，

```
10 D$="SCORE TIME LEFT":S
CORE=10:LEFT=102
20 MID$(D$,1,5)=RIGHT$("
 "+STR$(SCORE),5)
30 MID$(D$,7,4)=RIGHT$("
 "+STR$(TIME),4)
40 MID$(D$,12,4)=RIGHT$("
 "+STR$(LEFT),4)
50 PRINT D$
```

極めて明瞭かつ簡潔に表現ができる。

MID関数は指定した文字列の中から必要な文字列の取り出し，代入の機能をもち，RIGHT, LEFT両関数もほぼ同様の処理を行える。またSTR関数は数値を文字列に変換できるために，これらの組み合わせにより多彩な処理を行うことが可能である。

3-3　CHR$, HEX$, VAL, ASC

機械語を扱う場合，よくこの関数を用いる。マニュアルの付録を見るとアスキーコード表がある。例えばDは16進数でコードが44であるから，これをCHR$を用いて表示すると，PRINT CHR$(&H44)となり，このDというキャラクタが画面に表示できる。アスキーコード表右側には各種コントロールコードが並んでいる。これも画面表示と同様CHR$を用いてプリンタに直接コードが送ることができる。

また，ワードプロセッサのような文字列を扱うプログラムの場合，文字データをアスキーコード化して格納し，取り出すことが必要となる。**図1**ではLで示されるアドレスからX$の文字列を格納し，空白を1つ置いてLを次に進める。一例を次に示す。

```
10 LX=LEN(X$):X$=X$+" "
20 FOR I=1 TO LX:POKE L,A
SC(MID$(X$,I,1)):L=L+1:NE
XT:RETURN
```

このようにすると文字データは，JAPAN␣が16進数で4A, 41, 50, 41, 4E, 20と格納される。逆にLで示したアドレスから単語を1つ取り出し，X$に代入するプログラムの一例を次に示す(**図2**参照)。

```
10 X$=""
20 WHILE PEEK(L)>&H20:X$=
X$+CHR$(PEEK(L)):L=L+1:WE
ND:L=L+1:RETURN
```

(a)の例ではCHR$の逆関数であるASCを用いた。ASC関数は任意の文字のアスキーコードを与える。

3-4　ちょっと便利な機械語(16進数)入力プログラム

ストリング関数の結びとして，キーボードのテンキー類から16進データを入力するプログラムを紹介する。テンキーまわりの演算子キーでA～Fが入力でき，結構実用的である。このプログラムを実行するとスタートアドレスを聞いてくるので，16進で入力してから機械語データがストアできる。プログラム中，VAL("&H"+Y$)は&Hの付いてない16進データを読み込むためで，またRIGHT$は表示をそろえる工夫である。

```
10 X$="":FOR I=1 TO 16:RE
AD Y$:X$=X$+CHR$(VAL("&H"
+Y$)):NEXT
20 DATA 30,31,32,33,34,35
,36,37,38,39,2C,2B,2D,D,2
E,1C
30 INPUT "Address=";A$:A=
VAL("&H"+A$)
40 X=0:Y=INSTR(X$,INPUT$(
1)):IF Y=0 THEN 40 ELSE X
=(Y-1)*16
50 Y=INSTR(X$,INPUT$(1)):
IF Y=0 THEN 50
60 X=X+Y-1:PRINT RIGHT$("
000"+HEX$(A),4);"-";RIGHT
$("0"+HEX$(X),2):POKE A,X
:A=A+1: GOTO 40
```

FM-7の場合は，以下のように変更する。

```
20 DATA 30,31,32,33,34,35
,36,37,38,39,2A,2F,2B,2D,
3D,2C
```

4 知っておきたい基礎知識

使ってみれば実に便利であるけれど，気付かないことは結構あるものである。本体を入手すると数冊の厚い本（マニュアル）が付いてくるが，とても小説のように読破できるものではない。また，たとえすべて読んだとしても，それがオペレーション，プログラミングに関し，生かし切れるものではない。マニュアル製作側もそれなりの配慮，考慮のうえ編集しているのだろうが，製作者とわれわれユーザー（とくに初心者）との知識レベル差が問題であるように思う。いずれにせよ使わなければ無駄になるので，マイペースで付き合うことをおすすめしたい。

ここでは，F-BASICの仕様を説明する。マニュアルにも記載されているが，改めて取り上げるのはソフト上のいわゆる定格であり一応は知っておいていただきたいことであるからだ。

4-1 データの型

数の型は通常あまり意識せずにいても，支障なくプログラミングできる。F-BASICにおけるデータの型は**表2**のとおりで，変数名の後の記号（#，！など）により型が決定する。ただしDEF文をあらかじめ使って定義すれば，何の記号がなくとも各種の型にすることが可能である。DEF文は次の種類がある。

DEFINT（整 数）
DEFSNG（単精度）
DEFDBL（倍精度）
DEFSTR（文 字）

使い方の一例を次に示す。

```
10 DEFINT B
20 DEFSTR C-F
```

このようにすると，この後Bで始まる変数は，前述の型記号がない限り整数として扱かわれる。同様にC, D, E, Fで始まる変数は，文字型と見なされる。ただし，DEF文による型定義は前もって行う必要があり，そうでないと変数の値が初期状態に戻ってしまう。

このDEF文使用の利点の第1は，処理速度の高速化である。

```
10 TIME$="00:00:00"
20 FOR A=1 TO 10000:B=B+1
:NEXT:PRINT TIME$
```

上のプログラムに

```
5 DEFINT A-B
```

を追加して，比較すると一目瞭然である。

利点の第2は，メモリ領域の節約である。まず変数の記号の分だけ（正確には中間コード分）少なくて済む。さらに変数の領域を節約することができる。整数型として宣言すれば2バイトずつ節約するわけだ。BASICテキストとリンクしてオブジェクトを置いたりする場合，Out of Memoryに気を配る。また，高速化も含めてマルチステートメント化も有効な手段の一つである。

4-2 関数定義

ぜひ知っておいて使っていただきたい命令にDEF FN文がある。これはユーザーが自由に定義できる関数で，頻繁に使う数式はこの命令を用いて定義しておくと便利である。関数名はFNで始まるようにしなければならない。

4-3 条件分岐（IF文）

必ずプログラム中に出てくる命令の一つにIF文がある。一般的に，IF～THEN～ELSE～の型で用いるが，ネスティングを深めたり，マルチステートメントにすることにより構造化も可能で，また簡潔にもなる。しかし，GOTO文と同様便利なため，スパゲッティプログラム（GOTO文の多用により処理の流れがまるでスパゲッティのように混乱してしまうこと）にもなり得るので，使用時にはよく検討することが必要である。

5 エピローグ

富士通のFMシリーズはBASICに互換性があり，たいへん喜ばしい。オールインワン設計で売っている某社のBASICは，何と1機種でテープ／ディスクで8タイプ，また，国内実績No.1の某社でさえ，ソースレベルでのコンパチビィリティはなくなってきた。遅ればせながらJIS BASICが制定されたが，どれだけ採用されるか。いま唯一言えることは，主流がM-BASICであることは変わらないということだ（M-BASICが優れているということではない）。新製品が今後も出てくることは間違いないが，互換性を十分考慮しないとソフトウェアの蓄積はできない。言い換えれば良いソフトウェアには恵まれない。

F-BASICを限られた範囲で紹介してきたがいかがだろうか。本稿をお読みになり，少しでもBASICを理解していただけたのなら，このうえない喜びである。

ベーシック BASIC

山口 誠

富士通のマニュアルは非常に親切で，全くの初心者でも一通り読めばわかるようになっています。今までの日本製のパソコンのマニュアルで，これほど親切なものは見たことがありません。

と，これでは，BASIC講座に書くことがなくなってしまいますし，私も失業してしまいます。ところがどっこい，マニュアルを読んだだけではピンとこないところが結構あります。

皆さん，ファイルという概念をご存知ですか？　初心者の人の多くは，このファイルという概念が出てきたところでつまずくものです。OPENとか，CLOSE，PRINT#，INPUT#などの命令を使ったことがありますか？　これが使えないと，メモリ上のデータをテープやディスクに保存することができないんですね。この機会に理解しましょう。

テープとディスク

データを保存するものには，主に，テープレコーダとディスクがありますが，両者には，機能上大きな違いがあります。

カセットテープで音楽を聴く場合を考えてみましょう。このときテープの頭から聴く場合は問題ないのですが，例えば4曲目から聴きたいときは，自分，またはそのデッキが4曲目の頭を探し出して，それからその曲を聴くことになるわけで，すぐにというわけにはいきません。さらに，それが4曲目の第18小節目の三つ目の音を聞きたいなどとなったら，もうメチャメチャですね。

もし，これが音楽でなくコンピュータのデータならば，そういう細かい指定をしたい場合もあるわけです。

さて今度は，レコードで音楽を聴く場合を考えてみてください。テープのときと違って，4曲目からということは楽にできますね。さらに，ある一つの音が入っている場所さえわかれば，もっと細かい指定もできないことはありません。

もうおわかりでしょう。テープの場合は頭から順番に記憶して頭から順番に読み出すような方式に向いているのですが，任意の位置にデータを書いたり，任意の位置からデータを読んだりすることには向いていないのです。これに対し，ディスクの方は両方に向いているわけです。

頭から順に書いて，頭から順に読むように作られたデータの集まりをシーケンシャルファイルといい，任意の位置で読み書きができるように作られたデータの集まりをランダムファイルといいます。当然，テープの場合はシーケンシャルファイルのみ，ディスクの場合は両方とも作れるのです。

シーケンシャルファイル（テープ）

まず理由はともかくとして，一つのファイルを作る前には，必ずOPEN文を実行しなければならないということと，作り終えたときは必ずCLOSE文を実行しなければならないということを覚えてください。OPEN文を忘れる人は少ないのですが，CLOSE文は忘れがちなので気をつけましょう。

シーケンシャルファイルを作る場合，気をつけることは区切り記号についてです。これに気をつけないと，せっかく書いたデータが，あとでうまく読めなかったりするので，しっかり理解しておきましょう。

これを理解する一番の方法は，テープやディスクに実際何が書かれるかを考えることです。具体例を示しましょう。

```
10  OPEN "O", #1, "CAS0:TEST"
20  PRINT #1, "ABC": PRINT #1, "DEF"
30  CLOSE 1
40  END
```

上のプログラムを解説しましょう。まず行番号10のOPEN文ですが，これは「出力モードで，ファイル番号を1として，カセットにTESTというファイルを開け」という意味です。これだけ聞いても意味がさっぱりわからない人もいるでしょう。

まず「出力モードで」の意味ですが，これはシーケンシャルファイルを新たに作るときにつけるものです。これに対して，すでに作ってあるファイルを読み込むためにOPENするときは“I”を指定します(**表1**)。

表1　ファイルのモード

“I”	すでに作ってあるシーケンシャルファイルを読み込むためのモード
“O”	シーケンシャルファイルを新たに作るためのモード
“A”	すでに作ってあるシーケンシャルファイルの一番最後以降に，さらにデータを書き込むためのモード
“R”	ランダムファイルをオープンするためのモード

「カセットに，TESTというファイルを」はわかるでしょう。ほら，プログラムをSAVEするときも，SAVE“……”なんて書くでしょう。この“………”も，プログラムという一種のファイル名なのです。このファイル名によって，いろんなファイルを区別するわけですね。

さて，残った「ファイル番号を１として」ですが，これは少し難しいかもしれませんね。このプログラムの場合，OPENされているファイルは一つだけですが，ディスクやプリンタを使っているときなどは，よく二つのファイルが両方ともOPENされている，ということが生じる場合があります。しかし，ファイル番号という識別番号をOPENのときに決めてしまって，それ以降のPRINT＃文などのときに，＃のあとにその番号を書けば，どのファイルに出力するかが区別できるわけです。

例にあげたプログラムを見てください。行番号20, 30のPRINT＃文，CLOSE文には１という番号がついていますね。これはそれぞれ「１番としてOPENされたファイルに書き込め」，「それを閉じろ」という意味になります。

次の行番号20ですが，ここではABCという文字データと，DEFという文字データを１番のファイルに出力しています。ここで先ほど述べた区切り記号について考えてみましょう。まずこの20行を実行すると，結果として，テープに次のように記録されます。

……	A	B	C	⏎	D	E	F	⏎	………

ここで気をつけるのは，ABC，DEFのあとに改行を意味するコードが入っていることです。ほら画面にPRINTするときもPRINT“ABC”のあとに；をつけなければ，自動的に改行されますよね。これは，PRINT文が最後に自動的に改行コードを送っていると考えることもできるでしょう。つまり一つのPRINT文は，最後に；をつけないかぎり，自動的に改行コードを出してしまうわけです。ただ，この改行コードは画面では目に見えないため，普段は気にしませんよね。

これで20行の結果，どのようにテープに記録されるかはおわかりでしょう。

行番号30は先ほど忘れないようにと言ったCLOSE文です。これがないと，全てのデータが書き込まれないことがあり，あとで読み出すときにおかしなことになってしまいます。CLOSE 1を行うと，それ以降別のファイルに１番という識別番号を割り当てることができます。

さて，それでは実際テープにどのように書き込まれるかを，もう少し詳しく述べましょう。

先ほどの例では，ABCという文字列を出力しましたが，これが数値の場合はどうなるのでしょう。先ほどのプログラムの行番号20を消して，

```
20  A=30.5
25  PRINT#1, A
```

に換えると，

…………	␣	3	0	.	5	␣	⏎	…………

と記録されます。よく見てください。これは画面にPRINTしたときと同じ書式になりますね。

ここまではすんなり理解できたでしょう。しかし，少し考えなくてはならないのは，一つのPRINT＃文で複数のデータを書く場合です。

実はこの場合も，PRINT文が自動的に出力してしまう改行を意味するコードにのみ気をつければ，あとは普通のPRINT文と同じなんです。例えば，先ほどと同様に，

```
20  A=30.5：B=1000
25  PRINT#1, A；"ABC"；B
```

とすると，

…	␣	3	0	.	5	␣	A	B	C	␣	1	0	0	0	␣	⏎	…

のように記録されます。画面へのPRINT文と同じでしょう。

さて実際に記録される書式がなぜこんなに重要なのでしょうか。それは，これをINPUT＃文で読み込むときに問題が生じるからなのです。

普通のINPUT文を考えてみてください。INPUT文の場合はReturnキーが押されるまでキーボードから入力したものが，変数に代入されますね。しかしINPUT＃文は，キーボードからではなくて，ファイルから変数へ，区切り記号となる文字コードが入ってくるまで入力することになります。INPUT＃文を使うとき，とくに気をつけなくてはいけないのがこの区切り記号なんです。普通のINPUT文の場合，複数の変数に入力をする場合は，キーボードから打つときにカンマ（，）で区切らなければなりませんよね。しかし，INPUT＃文で数値を入力する場合は，，や；や改行のほかに空白文字（␣）も区切り記号になります。（文字の場合は，と；と改行のみ）（**表２**）。

表２　INPUT＃文における区切り記号

数値をインプットする場合	「空白」，「，」，「；」，「改行」
文字をインプットする場合	「，」，「；」，「改行」

では具体的に考えてみましょう。次のプログラムを見てください。

```
10  OPEN "I", #1, "CAS0:TEST"
20  INPUT #1, A, A$, B
30  CLOSE 1
40  END
```

このとき，

␣30.5␣ABC␣1000␣⏎

のように記録されたテープを読み込むと，どのようになるでしょうか。

A=30.5，A＄="ABC"，B=1000

となると思ったら間違いですよ。

先ほども言ったように，ファイルからの数値の入力のときは，空白も区切り記号になりますが，文字の入力のときはならないんです。だってそうでしょう。もし空白が区切り記号になってしまったら，空白を含む文字列はブッタ切りになってしまうでしょう。ですから結果は，

A＝30.5, A$＝"ABC␣1000␣"

となって，Bには何も入力できません。したがってエラーとなります。

それでは，どうしたらこの問題は解決するでしょうか。一番簡単なのは，一つのPRINT＃文では1つの変数しか出力しないようにすることでしょう。そうすれば，全てのデータのあとに，Returnキーに相当するコードが書かれますのでバッチシ！ です。

他の方法としては，次の方法があげられます。

PRINT＃1, A；"，"；"ABC"；"，"；B

この命令を実行した場合，次のようにテープに書き込まれます。

…	␣	3	0	.	5	␣	,	␣	A	B	C	,	␣	1	0	0	0	␣	⏎	…

おわかりでしょう。つまり強制的にカンマを入れてしまうんです。こうしておけばカンマのところで必ず区切られて入力されるでしょう。先ほど数値のときは空白も区切り記号になるといいましたが，考えるのが面倒な人は出力のとき，変数の間にとにかくカンマを入れてやってください。

さあ，これでテープベースでのシーケンシャルファイルの作り方はおわかりいただけたでしょうか。普通のINPUT，PRINTと違って，ファイルのときはデータの書式に気を使わなくてはなりません。

シーケンシャルファイル（ディスク）

テープにシーケンシャルファイルを作る作り方がわかれば，ディスクの場合も簡単です。ただ，ディスクの場合は，OPENの時に，

OPEN "O", 1, "0：TEST"

とするだけです。0：というのは，ドライブ0に，ということで，ドライブ1にしたいときは，1：にします。

```
10 OPEN "O", 1, "0：TEST"
20 PRINT＃1, "ABC"：PRINT＃1, "DEF"
30 CLOSE 1
40 END
```

これで，データはディスク上に，

のように書き込まれます。

このプログラムは，先ほどのテープの場合とほとんど同じですね。このプログラムで書き込んだデータを読み込んで，画面に出すプログラムを下に示しましょう。

```
10 OPEN "I", 1, "0：TEST"
20 INPUT＃1, A$：INPUT＃1, B$
30 PRINT A$：PRINT B$
40 CLOSE 1
50 END
```

とにかく，ディスクにシーケンシャルファイルを作るときは，ディスケットの中に，テープみたいなものがあると思って，テープのときと同様にやればいいわけです(**図1**)。

ランダムファイル

ランダムファイルの場合は，データを任意の位置で読み書きできるのですが，具体的にはどうすればいいのでしょうか。

それにはまず，レコードという概念を理解することから始めましょう。レコードというのは，256バイトのデータのかたまりと思ってください。

ところで，整数変数は2バイト，実数変数は4バイト，倍精度実数変数は8バイト，文字変数は1文字が1バイトの大きさであることは知っていますか？　知ってください。レコードとは，これらの変数を，目的によっていろいろ組み合わせて256バイト以内にしたデータのかたまりのことなのです。任意に読み書きできるというのは，この，データのかたまりの単位，つまり，レコード単位に任意に読み書きできるということなんです。ランダムファイルをオープンして，レコードナンバーと呼ばれるレコードの整理番号のようなものを指定すると，ディスケット内の256バイトの記憶部分が指定できるわけです。

とにかく，実例をあげましょう。

```
10 OPEN "R", 1, "0：TEST"
20 FIELD 1, 50 AS A$
30 LSET A$＝"ABCDEFG"
40 PUT 1, 1
50 CLOSE 1
```

図1

シーケンシャルファイル(テープ)　シーケンシャルファイル(ディスク)　ランダムファイル

このプログラムを解説しましょう。最初の行番号10はファイルのオープンですね。シーケンシャルファイルのときには，“O”や“I”が指定されていましたが，ランダムファイルのときは“R”です。ランダムファイルの場合は，1回オープンすると，任意のときに読み書きができますので，ファイルのオープンのときの入力/出力の区別は必要ないのです。

次のFIELD文ですが，ここで，1レコード内の変数の割り付けを行います。ここで示された変数に，後述するLSET，RSETという特殊な代入命令で書き込みたいデータを代入し，PUTという命令で一気にディスクに書き込むのです。ここでは，FIELD文で，50文字分の大きさの文字変数A$を1レコードに割り振っています。前に1レコードの大きさは256バイトと言いましたが，この場合，256−50＝206バイトはムダになっています。このFIELD文については，読みすすめていくうちに意味がわかっていくでしょう（**図2**）。

図2　フィールド文によるレコードの割り付け例

次は先ほど少しふれたLSET文です。FIELD文で示された変数は普通の変数と違って普通の代入文による代入ができないのです。理由はともかくとして，この変数は，ランダムファイルのために定義されたものなので，LSET，RSETでなければ代入ができないと覚えてください。先ほどのFIELD文でA$を50文字に設定しましたね。つまり，A$は50文字の大きさを持っているのです。そこに文字を左づめで代入するときはLSETを，右づめで代入するときはRSETを使います。

行番号40のPUT 1, 1は，「1番のファイルに（当然ランダムファイルである必要があります），レコード番号を1として，現在，メモリ上にあるレコード（この場合はA$ですね）を書き込め」という意味です。

レコード番号を1として，の意味ですが，このレコード番号とは，一つのランダムファイル内で，レコードを識別するための整理番号なのです。ランダムファイルでは，この整理番号によって，レコード単位で任意のデータを選択的に読み書きすることができるのです。当然，PUT 1, 1をPUT 1, 7にすれば7番レコードに書き込むことができるわけです（**図3**）。

そして，最後がCLOSE文です。これはシーケンシャルファイルのときと同様に，そのファイルを使い終わったら必ずつけなければなりません。

FIELD文について

さて，それでは，FIELD文を詳しく説明しましょう。先ほどは，

FIELD 1, 50 AS A$

としましたね。これは，1レコードを

というように割り振っていることを意味します。例えば，A$のあとに，さらに100文字分のB$を定義したいときは，FIELD文を，

FIELD 1, 50 AS A$, 100 AS B$

とやればいいのです。この場合，レコードは，

というようになります。この割り付けをうまくやれるようになると，ムダなあまり部分を少なくすることができるようになります。

数値を記憶させる方法

ここで重要なことが一つあります。前の例では，たまたま，A$とかB$とかの文字変数を使いましたが，実は，このレコードの割り付けには，文字変数しか使えないんです。つまり，ランダムファイルに記憶させることができるのは，文字変数だけなのです。

では，数値を記憶させるときはどうすればいいのでしょうか。要するに，ある数値変数があった場合，それと一対一に対応する文字変数があればいいんですね。それがちゃんとあるんです。つまり，

MKI$（整数値）＝整数を2バイトの文字変数とする関数

MKS$（実数値）＝実数を4バイトの文字変数とする関数

図3　ランダムファイルとレコード

MKD$(倍精度実数値)＝倍精度実数を８バイトの文字変数とする関数

CVI(２文字の文字列)＝MKI$で変換され，文字列となってしまった整数値を元の整数値に戻す関数

CVS(４文字の文字列)＝CVIと同様で，実数に戻す関数

CVD(８文字の文字列)＝CVIと同様で，倍精度実数に戻す関数

という一連の関数があります。ランダムファイル上では，ただ形式的に文字列として扱おうということだけですので，「とにかく，ランダムファイルで数値を扱うときは，これらの関数が必要なんだ」と憶えてしまってもかまいません。まだよく意味がわからない方がいらっしゃると思いますが，この先の実例を見ればおわかりになるでしょう。

```
10 OPEN"R",1,"0:TEST"
20 FIELD 1,2 AS A$, 4 AS B$, 8 AS C$
30 LSET A$=MKI$(82)
40 LSET B$=MKS$(9.18)
50 LSET C$=MKD$(3.1415926)
60 PUT 1, 1
70 CLOSE 1
```

ランダムファイルの読み込み

ここで，MKI$，MKS$，MKD$がそれぞれ２，４，８バイトの大きさであるのは，整数変数，実数変数，倍精度実数変数がそれぞれ２，４，８バイトの大きさであるためです。MKI$などを行った場合，数値がどんな文字になるかというのは考えなくてもいいでしょう。とにかく，何らかの文字列になるんです。

さて，これを読み込むときは，

```
10 OPEN"R",1,"0:TEST"
20 FIELD 1,2 AS A$, 4 AS B$, 8 AS C$
30 GET 1, 1
40 PRINT A$:PRINT B$:PRINT C$
50 CLOSE 1
```

ということになります。つまり，GET*n*, *m*を実行すると，*n*番のファイル(当然ランダムファイルです)の*m*番目のレコードを，FIELD文にしたがって，文字変数(この場合，A$，B$，C$ですね)に代入するということになるんです。

これまでの説明で，レコードに，どのようにデータを書き込むか，また，レコードからどのようにデータを読み込むかは，おわかりいただけたでしょう。これがわかってしまえば，多くのレコードを管理することもすぐにできます。

実　践　編

```
10 OPEN"R",1,"0:TEST"
20 FIELD 1, 20 AS N$, 100 AS A$, 15 AS T$
30 INPUT"ナンバー";X
40 IF X=0 GOTO 200
50 INPUT"ナマエ";NA$:LSET N$=NA$
60 INPUT"ジュウショ";AD$:LSET A$=AD$
70 INPUT"デンワ";TE$:LSET T$=TE$
80 PUT 1, X
90 GOTO 30
200 CLOSE 1
210 END
```

これは，レコード番号，名前，住所，電話番号を入力すると，そのレコード番号のところに，それらを記憶するというプログラムです。

行番号50，60，70で，何だか変なことをしていますね。一度，NA$，AD$，TE$という変数にINPUTしておいてから，それぞれN$，A$，T$にLSETで代入しています。これまでにも言ってきたように，FIELD文にある変数は，早い話が変数ではないのです。だから，こんな面倒なことになります。

さて上のプログラムを見ると，任意の位置に，の意味がよくわかるでしょう。逆に，任意の位置から，というプログラムもすぐにできるでしょう。

```
10 OPEN"R",1,"0:TEST"
20 FIELD 1, 20 AS N$, 100 AS A$, 15 AS T$
30 INPUT"ナンバー";X
40 IF X=0 GOTO 200
50 GET 1, X
60 PRINT N$:PRINT A$:PRINT T$
70 GOTO 30
200 CLOSE 1
210 END
```

さて，これで大体はおわかりいただけたと思います。これで一応，用は足りますので，ここまで理解するだけでも十分でしょう。ただ，どんな短いデータにも256バイトを使ってしまうというのはムダだと思いませんか？　これを解決する方法はいろいろあるのですが，ここでは，私が愛用している方法を一つお教えしましょう。

ムダなくレコードを割り付ける法

例えば，最大20文字にしかならないような人の名前ばっかりを，いっぱい記憶させたいときはどうすればいいのでしょうか？　これを今までのやり方でやれば，20文字しか必要がなくても，256文字分使ってしまいます。でも，次のようにすればムダが少なくなります。

```
10 OPEN"R",1,"0:TEST"
20 FIELD 1, 240 AS A$
30 INPUT"ナンバー";X
40 IF X=0 GOTO 200
50 X1=(X-1)¥12+1:X2=((CX-1)MOD12)*20
60 GET 1, X1:C$=A$
70 INPUT B$:B$=LEFT$((B$+SPACE$(20), 20)
80 LSET A$=LEFT$(C$, X2)+B$+RIGHT$(C$, 220-X2)
```

```
 90  PUT 1, X1
100  GOTO 30
200  CLOSE 1
210  END
```

ランダムファイルを初めて習う人にとっては，ちょっぴり難しいかもしれませんが，がんばって学習してください。この場合，1レコードに12人分の名前を入れています。その通し番号がX，その番号の人の名前を入れるべきレコード番号がX1，そのレコードの中で，何文字目の次からかを表すのがX2です。X1は(X-1)を12で割ったときの整数部分，X2は，(X−1)を12で割ったときのあまりの20倍になります。その計算をしているのが50行です。それをLSETする際，LEFT$，RIGHT$を使って，正しい位置に名前を入れていますね。こうすれば，一つのレコードの中に多くのデータを書き込めます(**図4**)。

図4　レコードを効率よく使う法の一例

A$ | 20文字 | 20文字 | 20文字 | | 20文字 | あまり |

0×20＋1文字目　1×20＋1文字目　2×20＋1文字目　11×20＋1文字目　12×20＋1文字目

A$

レコード番号													
1	1	2	3	4	5	6	7	8	9	10	11	12	
2	13	14	15	16	17	18	19	20	21	22	23	24	
3	25	26	27	28	29	30	31	32	33	34	35	36	
4	37	38	39	40	41	42	43	44	45	46	47	48	
5	49	50	51	52	53	54	55	56	57	58	59	60	
6	61	62	63	64	65	66	67	68	69	70	71	72	

たとえば，43番のデータを見たい時は，
レコード番号＝(43−1)￥12＋1＝4
何文字目の次か？＝((43−1)MOD12)＊20＝120
だから
MID$(A$, 121, 20)
とすれば：43番目の20文字データが与えられます。

このほかに，FIELD文を何回かに分けて，その中で，文字配列を使い，上のX2にあたるものを配列の添字にするという方法もあります。どちらがいいかは，その場その場でよく考えてみてください。

ランダムファイルをもっとよく理解したい方は，各自で，自分なりに住所録のようなものを作ってはどうでしょうか。とにかくコンピュータは習うより慣れろです。どんどんプログラムを作ってみましょう。

★耳よりな情報★

マイコンショウ'83
5月25日から開催

(社)日本電子工業振興協会主催のマイクロコンピュータショウ'83が，5月25日(水)〜28日(土)の4日間，東京流通センター展示場（平和島）で開催される。

今年のテーマは"知能化時代とマイコンの役割"。ハード，ソフトメーカー114社（昨年は105社）が参加出展する。注目の富士通ブースは1−7。ちなみに，本誌の日本ソフトバンクも参加します。ブースは1−17。

役に立つプログラムを作るための

ソフトウェア設計法入門講座(3)

Computer Science Group　武原　宰

IBM社のパーソナルコンピュータが，いよいよ日本にも上陸した。昨年，アメリカで発表されるや，たちまちアメリカのビジネス用パソコン市場のトップシェアを確保したIBM-PCである。

アメリカ版がメインＣＰＵに8088を用いているのに対し，今回，日本で発表されるマルチステーション5550は，8086を採用している。

日本の各メーカーの優れたハードウェア技術に対抗するため，アメリカ版よりも一段グレードアップした純16ビットマシンを引っさげての上陸である。

処理能力はアメリカ版の３倍以上，日本語処理機能も盛り込まれ，ワープロとしても，漢字端末としても活用できるという高性能パソコンである。日本のビジネス用パソコン市場に一大旋風を巻き起こすことは間違いないだろう。

ところで，なぜIBM社はコンピュータ業界の巨人と呼ばれ，超大型機からパソコンに至るまで第一人者の座を占められるのだろう。

ハードウェア面に関しては，日本の各メーカーは，その優れたIC製造技術により，世界の最高水準にあるといえる。ところが，ソフトウェアに関しては，IBM社の長年にわたる蓄積にどうしても追いつけないのが現状である。

これは，ハードウェアの製造が，大量の資金を投じて優れた製造設備をそろえれば，それなりの製品をつくることができるのに対し，目に見えないノウハウの集まりであるソフトウェアでは，その蓄積の大部分が人間（プログラマ）の頭の中だけに残りやすく，よほどの努力をしないかぎり，他人が利用し得る形で外部にアウトプットされないことに依っている。

IBM社では，このプログラムを他人が利用できる形に変えて残す，という最も困難な課題に長年取り組んできた。そして，その結果として他社が容易に追従できない豊富なソフトウェアの蓄積を実現したのである。

この講座では，IBM社で開発されたソフトウェア設計法の一つであるHIPO法について解説してきたが，この手法自体，プログラムの再利用と蓄積を目的として考案されたものなのだ。

個人的な業務や研究にパソコンを活用する場合にも，プログラムの蓄積，すなわち以前につくったプログラムの再利用ということが，効率のよいプログラム開発を行ううえでの，最も重要なキーポイントとなる。

今回は，再利用頻度の最も高いサブルーチンの設計法と文書化について考えてみることにしよう。

I. サブルーチンの整備

自分の作成したプログラムでも，１か月もたつとすっかり忘れてしまうのが普通である。そのプログラムを再利用しようとする場合，ドキュメントが何もないと，リストを全部読み直し理解するしか方法がない。

また，ドキュメントがあっても，そのプログラムが特殊な用途専用につくられており，他の仕事に利用しようとすると大幅な手直しが必要で，新しくつくり直した方が早い場合もある。

プログラムの再利用を行おうとする場合には，何よりもまず，汎用性のあるプログラムをつくり，その使用説明書をきちんと整理しておくことが大切だ。

汎用性のあるプログラム，すなわち使用頻度の高いサブルーチンをつくるためにまず大切なことは，どのようなサブルーチンをそろえるかということである。サブルーチンをその機能から分類し，どのような体系でサブルーチンを蓄積していくかということを，最初に充分考えておく必要がある。

サブルーチンは，プログラムという全体を構成する基本的な部品である。家をつくる場合や，車をつくる場合を思い浮かべてもわかるように，部品はあまり大きすぎてはいけないし，また，あまり小さな部品ばかりでもいけない。適度の大きさの部品が，必要な種類だけそろっていて，はじめて家なり車なりの全体を効率よくつくりあげることができる。

部品の種類が豊富でかつ，体系立ってそろっているほど，いろいろなバラエティに富んだ全体を，すばやくつくりあげることができるのだ。サブルーチン整備の方法もこれと同じである。体系立って部品集めをしていくことが大切なのだ。

Ⅱ.サブルーチンの分類

プログラムをその機能面で分類しようとした場合は，まず,「入力」「処理」「出力」に大きく3分割してみることが，一番確かな方法だ。これは，コンピュータそのものが，データを入力して，処理し，出力するように働く装置であるため，それを動かすプログラムも当然，その働きに従って分類すると間違いないのである。

さらに，各「入力」「処理」「出力」の中を分類すると，**図1**のようになる。「入力」や「出力」は必ずその対象となる装置が存在するので，その装置単位に分類しておけば，あとで使いやすくなる。つまり，このように分類しておけば，例えば，プリンタとディスクの両方がないと使えないサブルーチンなどというものは存在しないことになり，中途半端な汎用性のないサブルーチンの発生を未然に防ぐことができる。

一方,「処理」は，コンピュータにやらせようとする仕事の内容によって，項目が大きく変わってくるだろう。例えば，データベース的な処理であれば，ソートやマージなどという項目が，また図形処理であれば，二次元Affin変換や三次元透視変換などという項目が並ぶことになる。いずれにしても重複がないよう，また処理が大きくなりすぎないよう分類するのがコツである。

図1　サブルーチンの分類

Ⅲ.名前の付け方と行番号の割り付け

分類が決まると，次は一つ一つのサブルーチンに簡潔な名前を付ける。名前は，処理内容を端的に表現した日本語名と，英数字より成るファイル名（ディスク内あるいは，カセットテープ内でのファイル名称）の2つを考える必要がある。とくに，ファイル名称はサブルーチンの管理上，重要な意味をもつため，体系立った命名法をあらかじめ決めておくのがよい。筆者が用いている方法を紹介すると次のとおりである。

まず,サブルーチンの行番号として,1000～9999行までを割り当てる。サブルーチン一個当たりの行番号は最大100とし,その先頭行の10位および1位は必ず0とする。つまり，各サブルーチンは1000行，1100行，1200行という行から始まることになる。そして，個々のサブルーチンの名称は，サブルーチンの頭文字をとった「S」と，その先頭行番号2文字をとった「S××」(××＝先頭行番号の1000位と100位)という形で必ず付けることにする。したがって，ファイル名は，S10，S11，S12，……，S99というように一義的に決まることになる。

さらに，前述したサブルーチン分類と対応づけて，例えば，1000行台は「キーボード入力」,7000行台は「CRT出力」などと決めておくと，後々扱いやすくなる。さらにこの方法の最も優れている点は，サブルーチンを修正して，再度ディスクにセーブしようとするとき,必ず現在表示されているリストの行番号と，セーブしようとするファイルの名称とが，一致していることを確認できるため，誤ってファイルを壊すことがない，という点にある。セーブするファイル名を間違えて，大切なファイルを壊してしまうことがよくあるから，この利点はなかなか大きい。

Ⅳ.サブルーチン説明書の書き方

説明書は簡潔で，しかも必要なことは全て書かれていることが望ましい。そのためには，説明書に書くべき項目をあらかじめ定め,サブルーチンをつくるつどに,面倒がらず，決めた項目を全部埋めたサブルーチン使用説明書を書くことである。この努力がそのままソフトウェアの蓄積につながる。プログラムだけつくって説明書を書かなければ，しばらく時間がたつと何もつくらなかったのと同じことになってしまう。

筆者の使っている説明書の項目は，**図2**のとおりである。

このうち，コーリングシーケンスとは，サブルーチンをコールする際の手順のことであるが,BASICの場合はすべてGOSUB＋行番号という形式となる。ただし，サブルーチンにジャンプする前に必ず引数のセットがある場合や,サブルーチンでREADするDATA文の先頭を指定するため RESTORE文を実行する必要のある場合は，その説明を加えておくとわかりやすくなる。

また,内部変数は,サブルーチンを使用する面からだけでは必要ないが，サブルーチン内部の動作を理解する場合や，外部で使

用している変数との重複がないかを調べる場合に有効である。とくに後者の変数の重複の問題は，よく発生するバグである。

V. 変数名の決め方

上述したような変数名の重複を避けるためには，あらかじめ変数の命名法を定めておくとよい。筆者は**図3**のような方法を用いている。

この約束を守るかぎり，変数の重複によるトラブルはまず発生しない。読者の皆さんも，自分なりの理解しやすい約束事を工夫してみて欲しい。

図2　サブルーチン説明書の項目

名称		ファイル名	

1. 処理内容
   簡潔に，しかも漏れなく具体的に処理内容を記述する。図による表現を用いるのも望ましい。
2. コーリングシーケンス
   GOSUB ×××××
3. 入力変数
   サブルーチンに引き渡す変数名およびそのセット内容を記述する。
4. 出力変数
   サブルーチンより返される変数名およびそのセット内容を記述する。
5. 内部変数
   サブルーチン内部で使用する変数名およびそのセット内容を記述する。

図3　変数命名法

1. 原則として3文字とする。
   (長い変数名は，ひと目で理解しにくく，また，キー入力時に誤りやすい)
2. サブルーチン内部でのみ使用する変数は，末尾を必ず「S」とする。
3. サブルーチンおよびメインルーチンで共通して使用する変数は，先頭をZとする。
4. メインルーチンでのみ使用する変数は，2および3以外のタイプとする。
5. サブルーチンからさらにコールされるサブルーチンの内部変数は，末尾にサブルーチンファイルネーム（S××）を付加する。

リスト　例1

```
1500 '******************************************
1505 '*                                        *
1510 '*  S15 ( Message  Display Subroutine )   *
1515 '*                                        *
1520 '******************************************
1525 READ IS
1530 FOR JS=1 TO IS
1535     S$=""
1540     READ CS,XS,YS,NS
1545     COLOR CS
1550     LOCATE XS,YS
1555     IF ZB$(JS)<>"" THEN S$=RIGHT$(ZB$(JS),NS)
                        ELSE FOR KS=1 TO NS:S$=S$+"■":
                             NEXT KS
1560     PRINT S$;
1565     S=NS-LEN(S$)
1570    IF S<0 THEN S=0
1575    PRINT SPACE$(S)
1580 NEXT JS
1585 RETURN
```

リスト　例2

```
3900 '******************************************
3902 '*                                        *
3904 '*  S39 ( Line & Box Display Subroutine ) *
3906 '*                                        *
3908 '******************************************
3910 READ S$
3912 IF S$="E" THEN RETURN
3914 IF S$="B" OR S$="BW" THEN 3952
3916 '** Line**'
3918    READ CS,XS,YS,LS,DS,NS
3920    XDS=XS*8+3:YDS=YS*8+3:DDS=DS*8:LDS=LS*8
3922    IF S$="Y" OR S$="YW" THEN 3938
3924      '** Horizontal **'
3926        FOR IS=1 TO NS
3928             LINE (XDS,YDS)-(XDS+LDS,YDS),PSET,CS
3930             IF S$="XW" THEN LINE(XDS,YDS-1)-(XDS+LDS,YDS-1),PSET,CS
3932             YDS=YDS+DDS
3934        NEXT IS
3936   GOTO 3910
3938   '** Vertical **'
3940       FOR IS=1 TO NS
3942            LINE (XDS,YDS)-(XDS,YDS+LDS),PSET,CS
3944            IF S$="YW" THEN LINE(XDS-1,YDS)-(XDS-1,YDS+LDS),PSET,CS
3946            XDS=XDS+DDS
3948       NEXT IS
3950   GOTO 3910
3952 '** Box **'
3954   READ CS,X1S,Y1S,X2S,Y2S
3956   LINE(X1S*8+3,Y1S*8+3)-(X2S*8+3,Y2S*8+3),PSET,CS,B
3958   IF S$="BW" THEN LINE(X1S*8+1,Y1S*8+2)-(X2S*8+5,Y2S*8+4),PSET,CS,B
3960   GOTO 3910
```

VI. サブルーチンの例

最後にサブルーチンの例および使用説明書の例を示そう。いずれもCRT出力用サブルーチンである。

リスト例1はＣＲＴの指定座標位置に指定色で文字データを表示するものである。文字データが空のときには，指定文字数分の"■"マークを表示する。また，文字データの個数が指定個数以下のときは，指定個数分となるまで，スペースを出力するよう

になっている。これらの機能は，CRT画面上に表形式の表示を行う際に威力を発揮する。

リスト例2は，CRT画面上に格子を発生するサブルーチンである。格子の表示位置や間隔，本数，色などが自由に指定できるほか，二重線，二重ワク等の指定もできる。さらに便利なことには，表示位置をキャラクタ座標で指定できるため，前述のサブルーチンと容易に組み合わせて，格子枠付きの表をCRT画面上に自由に書くことができる。

しかも，これらのサブルーチンに共通しているのは，入力パラメータの引き渡しにDATA文を用いていることである。DATA文を用いることにより，表示位置，表示色，表示文字数等の情報をロジック部より切り離し，独立に管理することができる。つまり，それらのDATA文に特定の行番号を与えて集めておけば，画面のフォーマットを変更しようとした場合でも，サブルーチンの内容をいっさい見ることなく，DATA文にのみ着目してその値を変えることにより，自由にフォーマットを変更できる。

これらのサブルーチンは，いわば，BASICを用いて，自分自身の仕事に最も適した新しい言語をつくったようなものである。このように自分の仕事に適したサブルーチンを多数蓄積することによって，最初からすべてのルーチンをつくった場合に比べて，数倍から数十倍のプログラム開発力を発揮することができるようになる。IBM社のもつ底力も，このようなサブルーチンの蓄積のおかげであると言える。

サブルーチン説明書例1　CRTメッセージ出力

名称	CRTメッセージ出力	ファイル名	S15

1. 処理内容

DATA文により与えられる各メッセージごとの表示開始XYキャラクタ座標，表示色コード，表示文字数に従い，所定エリアにセットされた文字データをCRT画面上に表示出力する。

2. コーリングシーケンス

3. 入力変数

(1)DATA文

DATA　$m, c_1, x_1, y_1, n_1, c_2, x_2, y_2, n_2, \cdots\cdots, c_i, x_i, y_i, n_i, \cdots\cdots, c_m, x_m, y_m, n_m$

m：メッセージ個数（$1 \leq m$）

c_i：第iメッセージ表示色コード（$0 \leq i \leq 7$）

0…黒	4…緑
1…青	5…空色
2…赤	6…黄
3…紫	7…白

x_i：第iメッセージ表示開始X座標(キャラクタ座標)

y_i：第iメッセージ表示開始Y座標(キャラクタ座標)

n_i：第iメッセージ表示文字数

(2)表示用文字データ

ZB$($i$)：第$i$メッセージ用文字データ（$1 \leq i$）

4. 出力変数

なし

5. 内部変数

(1)変数

CS：表示色コード　XS：表示開始X座標

YS：表示開始Y座標　NS：表示文字数

IS：メッセージ個数

(2)カウンタ

JS：メッセージ個数　KS：文字個数

(3)ワークエリア

S$，S

サブルーチン説明書例2　CRT格子出力

名称	CRT格子出力	ファイル名	S39

1. 処理内容

DATA文により与えられる線種指定(水平線，垂直線，箱)，始点XYキャラクタ座標，終点XYキャラクタ座標，表示色コード，繰り返し回数，間隔などを得て，CRT画面上に，水平線，垂直線，箱を表示出力する。

2. コーリングシーケンス

```
    RESTORE  n
    GOSUB    3900
      ⋮
n   DATA     ......
```

3. 入力変数

(1)DATA文

a. 線種が「箱」の場合

DATA　$m_1, c_1, xs_1, ys_1, xe_1, ye_1, m_2, c_2, xs_2, ys_2, xe_2, ye_2, \cdots\cdots, m_i, c_i, xs_i, ys_i, xe_i, ye_i, \cdots\cdots$, E

m_i：第iボックスのモード（文字データ）

B…箱　BW…二重箱

(xs_i, ys_i)　(xe_i, ye_i)

c_i：第iボックスの色コード($0 \leq c_i \leq 7$)

xs_i：第iボックスの開始X座標(キャラクタ座標)

ys_i：　〃　開始Y座標(　〃　)

xe_i：　〃　終了X座標(　〃　)

ye_i：　〃　終了Y座標(　〃　)

E：エンドマーク(最終DATA文の末尾に必ず入れる)

b. 線種が「水平線」または「垂直線」の場合

DATA　$m_1, c_1, xs_1, ys_1, \ell_1, d_1, n_1, m_2, c_2, xs_2, ys_2, \ell_2, d_2, n_2, \cdots\cdots, m_i, c_i, xs_i, ys_i, \ell_i, d_i, n_i, \cdots\cdots$, E

(xs_i, ys_i)　d_i　n_i　ℓ_i

m_i：第iラインのモード(文字データ)

X…水平線　Y…垂直線　XW…二重水平線　YW…二重垂直線

c_i：第iラインの色コード($0 \leq c_i \leq 7$)

xs_i：　〃　開始X座標(キャラクタ座標)

ys_i：　〃　Y　〃（　〃　）

ℓ_i：　〃　長さ

d_i：　〃　間隔

n_i：　〃　繰り返し回数($1 \leq n_i$)

E：エンドマーク(最終DATA文の末尾に必ず入れる)

(注1)　1つのDATA文で，a.とb.の場合が混在してもかまわない。

(注2)　線および箱の始点・終点ドット座標は，始点・終点キャラクタの中心となる。

4. 出力変数　なし

5. 内部変数

(1)変数

CS：表示色コード　XS：ライン始点X座標　YS：ライン始点Y座標

LS：ライン長さ　DS：ライン間隔　NS：繰り返し回数

X1S：ボックス始点X座標　Y1S：ボックス始点Y座標

X2S：　〃　終点　〃　Y2S：　〃　終点　〃

(2)カウンタ

IS：繰り返し回数

(3)ワークエリア

S$，XDS，YDS，DDS，LDS

ケーススタディ――

ゲームばかりがパソコンじゃない！

FMシリーズ用PERS-F1~F2を建築の透視図に活用

山城デザインスタジオ

「パソコン＝ゲーム」では情けない，何とかパソコンを有効に活用したい，パソコンを使ってどんなことができるのか，どんな活用の仕方があるのか，と考えている読者も多いことだろう。この欄では，ＯＡ（オフィスオートメーション），ＦＡ（ファクトリオートメーション），ＬＡ（ラボラトリオートメーション），ＨＡ（ホームオートメーション）として，ＦＭがどんなふうに使われているのか，事例を紹介していく。

ＣＧ（コンピュータグラフィックス），３次元グラフィックスもご多分に漏れず，ゲームなどによく使われているが，ＦＭは，その優秀なグラフィック機能ゆえに，実務面で使われている例も多い。今回は，３次元グラフィックスを建築の完成予想図などに応用している山城デザインスタジオを紹介しよう。

山城社長

東京・恵比寿駅から歩いて３分ほど，明治通りに面したビルの中にある山城デザインスタジオは，建築パース制作の専門会社。パース（Perspective）とは，平面図や立面図から，透視図法という一定の原理にしたがって遠近感のある立体図におこしたもので，ビルやマンションなどの建物，造園や都市計画の完成予想図をビジュアルに表現したもの。「昔は物が建てばよかったが，今はどういうデザインのものができて，どういう空間・環境になるのか，という視覚的なものが要求される時代になってきた」と山城社長が語ってくれたように，建築プレゼンテーションとしての需要は，ますます多くなるだろう。

このパースをつくる仕事のなかで，平面図などから立体図に書き換えるところをパソコンにやらせている。「パースという仕事にはデジタル的なところとアナログ的なところと二面ある。デジタル的なところ，つまり透視図法を使って透視図を書く段階では，コンピュータが有効です。着彩など感性の必要なところは人間の感覚で表現しなければならない」と山城社長は分析してくれた。

このソフトは，山城社長が開いているパースとデザインの専門学校「東京パース造形学院」の講師でもある岡田氏が開発部長をしている㈱マール社プランニング・センターが，山城デザインスタジオ用に開発したもので，FM-8用の「PERS-F1」，FM-7用の「PERS-F2」があり，FM-11用の「PERS-F3」も近く完成の予定という。

図1のような立面図と平面図から，**図2**のような立体図におこすわけだが，従来のように人間が図面を読んで透視図法に基づき定規で線を引いていては，ベテランでも３～４時間かかる。PERS-F1を使って入力すれば２時間ぐらいでできあがる。しかも，視点を変えて**図3**のようなバリエーションを出すには，それぞれ５分くらいでOK。手作業では，それぞれにまた３～４時間はかかってしまう。一つの図面を入力すれば，数値を置き換えるだけで視点の移動が簡単にできる，というコンピュータならではの業である。CRT上に出したこうしたバリエーションの中から，顧客にどの視点から見たものを描けばよいか選んでもらう。決定した透視図をプロッタに出力し，それを基にして，彩色したパースを完成させる。完成したパースは建築予想図として建設中の建物の看板やマンションなどの広告に使われている。

PERS-F1などのプログラムはBASICで書かれている。「本当はマシン語で動かしたいんですが……」とオペレータの一人は話してくれた。BASICでは「心持ち」遅いと言う。

オペレートは，入力の方法さえ勉強すれば，透視図法などを学んだ専門家でなくとも簡単にできる。山城社長は，「作図法というのは，結局，単純作業ですからね。単純作業というものは機械がとってかわるものと考えています」と語る。

パソコン導入にあたっては，普通いくらかの不安を伴うものだが，「こうした学校をやっていると，新しいものを道具として使うことを勉強していく。不安はまったくなかった。むしろそうした道具をいかに有効に使うかが問題だ。コンピュータは命令どおりに動く。どういった命令をどう有効に使うかだ。パソコンレベルでは，すべてをコンピュータにまかせるところまではいっていない。コンピュータの得意な点をいかに生かすかです。」

「パソコンのいいところは，パーソナルな単位で身近に使える，簡単にさわれるという点です。これからは，おそらく建築をやる学生は，図面を引いてパースも自分自身，CRTで確かめるというのが当たり前の時代になるんではないでしょうか。」

数あるパソコンの中から，なぜFMシリーズを選んだのかとの質問には，「3次元パースには富士通の製品が使いやすいとのことで」と答えが返ってきた。FMのグラフィック機能が評価され，有効に使用されている。

お話を伺った東京パース造形学院の教室には，FM-7とCRT，ミニフロッピディスクユニット(シングル)のセットが5台，同11のセットが1台，それに岩崎通信機のパーソナルプロッタSR-6602，渡辺測器のマイプロットWX4675が各1台置いてあった。「これから，地方の方を集めて，集中セミナーを開くことになっています。それで，今日これからFM-7が11台入る予定です。富士通のPRをしてあげるようなものです」と笑っておられた。

去年10月頃に開発したというPERS-F1は，市販もされているが，「実用に使っているのはうちぐらい。まだ知られていないし，使い方もわからない」というわけで，同セミナーが開かれる。

「将来は建築プレゼンテーションの新しいメディアとして，映像プレゼンテーションの時代が来るんではないでしょうか。現在は平面図と模型やパースなどの立体図だけですが，CRTの精度向上，色彩などの改良などが図られれば，現場の風景などのビデオとコンピュータ映像とを合成した新しい映像プレゼンテーションが可能になる。合成画像を回転させることによって，その空間へ入って体験することができるでしょうし，また建物の外からドアを開けて入り，中の各部屋も体験できるという時代になってくるでしょう」と将来への展望を語ってくれた。

編集部よりお願い

家庭・ビジネス・工場などで，FMを有効に利用している方，またその例をご存知の方，ご連絡ください。ケーススタディとして取りあげ，読者に紹介いたします。パソコンをただの置物にしてしまっている人も大勢います。より多くの方が，より有意義に活用できるように，ぜひご協力ください。自薦・他薦を問いません。編集部あてに電話・ハガキなどでご連絡ください。

図2（プロッタで出力したものに陰線処理をしたもの）

図3（プロッタで出力したものに陰線処理をしたもの）

FM-7, 8, 11

三次元パッケージのアルゴリズム ワイヤフレームによる例

西村義孝

ワイヤフレームによる三次元パッケージのアルゴリズムを説明しよう。

最近のコンピュータグラフィックスは，物体を面のデータで扱うサーフィスモデルや，基本的な物体（立方体，円柱，円すいなど）の合成で物体を表現するソリッドモデルが中心で，辺のデータで物体をあらわすワイヤフレームはそれらのモデルを表示する補助的な手段として使われているのが現状といえる。ただ，ワイヤフレームによる三次元出力はかなり簡単なアルゴリズムで行えるため，全部あるいは一部をハードウェアで組んでリアルタイムに表示可能としているシステム（サイラックなど）も多いようだ。

ワイヤフレームの問題点は，人間が実際に見て感じる感覚とかなり異なってしまう点だろう。たとえば**図1**のようなワイヤフレームの図形があたえられると，**a**のようにも**b**のようにも**c**のようにも見える。奥行きの情報も欠落しているため**d**なのか**e**なのかもはっきりしない。遠くの物体ほど暗くするデプスキューという方法で，奥行きを補助することもあるが，それでも私たちの感覚からはかなりずれている。見てわかりやすいのは，やはりサーフィスモデルとかソリッドモデルだろう。

図1

これらでは反射，影，透過光，スムージングなどで，リアルに図形を表現することができる。が，反面，プログラムは複雑となり，大型コンピュータでもリアルタイムに動かすのは難しい。パソコンでサーフィスモデルやソリッドモデルを扱うこともできなくはないが，非常に時間がかかる。またシェーディング（物体の色の計算）を厳密に行ったとしても，パソコンの8色とか16色とかいった貧弱な色分解機能では，それを表現することもできないことになる。

パソコンではワイヤフレームによる表示が分相応といえるだろう。

三次元パッケージ

三次元パッケージの主要部分は，次の3つに分かれている。

1．座標変換
2．クリッピング
3．透視変換

それぞれを説明していこう。

1. 座標変換

座標には見る物体の属するワールド座標系と，見る人に属するビュー平面座標系とがある。ワールド座標系での物体のデータをビュー平面座標系に直さないと，その物体がどう見えるかがわからない。その変換が座標変換だ。ワールド→ビュー平面の座標変換は，平行移動と回転を組み合わせて行われる。

平行移動

話を簡単にするために二次元で考えてみよう。**図2**を見てほしい。

点Pはワールド座標系では（1，1）だが，ビュー平面座標系では（4，3）だ。これは，ワールド座標系がビュー平面座標系の（3，2）に平行移動しているからで，

図2

（4，3）＝（1，1）＋（3，2）

の関係が成り立つ。ワールド座標系の座標を (x_w, y_w)，ビュー平面座標系の座標を (x_v, y_v)，平行移動量を $(\Delta x, \Delta y)$ とすれば，

$$(x_v, y_v) = (x_w, y_w) + (\Delta x, \Delta y)$$

という関係式が成り立つ。三次元の場合は同様に

$$(x_v, y_v, z_v) = (x_w, y_w, z_w) + (\Delta x, \Delta y, \Delta z)$$

となる。マトリクス表現では同次座標表現*で

$$(x_v, y_v, z_v, 1) = (x_w, y_w, z_w, 1)T$$

$$\text{, where } T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \Delta x & \Delta y & \Delta z & 1 \end{pmatrix}$$

となる。

* 同次座標表現とは，平行移動，回転を同じように扱えるように (x, y, z) の座標にスケーリングを加えた $(x, y, z, 1)$ という形で扱うもの。

回転・回転移動

これもまず二次元で考えてみよう（**図3**）。点Pはビュー平面座標では（2，3），ワ

図3

ールド座標では複雑だが，$\left(\frac{2\sqrt{3}+3}{2}, \frac{-2+3\sqrt{3}}{2}\right)$

となる。これはワールド座標系がビュー平面座標系に対して30°傾いているためで，

$$(2,3)=\left(\frac{2\sqrt{3}+3}{2}, \frac{-2+3\sqrt{3}}{2}\right)\begin{pmatrix}\cos 30^\circ & \sin 30^\circ \\ -\sin 30^\circ & \cos 30^\circ\end{pmatrix}$$

という関係になる。一般に，ワールド座標がθ 回転している場合には

$$(x_V, y_V)=(x_W, y_W)\begin{pmatrix}\cos\theta & \sin\theta \\ -\sin\theta & \cos\theta\end{pmatrix}$$

となる。

三次元の場合もまったく同じだが，回転させる軸が問題となる。二次元の場合は1本しかありえなかったが，三次元の場合は無数に存在する。普通はx軸まわり，y軸まわり，z軸まわりの回転を考える。

x軸まわりの回転は回転角をφとすると

$$(x_V, y_V, z_V, 1)=(x_W, y_W, z_W, 1)R_x(\varphi)$$

$$\text{, where } R_x(\varphi)=\begin{pmatrix}1 & 0 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi & 0 \\ 0 & -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1\end{pmatrix}$$

y軸まわりの回転は回転角をθとすると，

$$(x_V, y_V, z_V, 1)=(x_W, y_W, z_W, 1)R_y(\theta)$$

$$\text{, where } R_y(\theta)=\begin{pmatrix}\cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1\end{pmatrix}$$

z軸まわりの回転は，回転角をψとすると

$$(x_V, y_V, z_V, 1)=(x_W, y_W, z_W, 1)R_z(\psi)$$

$$\text{, where } R_z(\psi)=\begin{pmatrix}\cos\psi & \sin\psi & 0 & 0 \\ -\sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1\end{pmatrix}$$

このx軸まわり，y軸まわり，z軸まわりの回転を合成して，はじめて三次元の回転が記述できる。最終的にx軸まわりにφ，y軸まわりにθ，z軸まわりにψ回転するには，

$$(x_V, y_V, z_V, 1)=(x_W, y_W, z_W, 1)R(x, y, z)$$

$$\text{, where } R(x, y, z)=R_x(\varphi)R_y(\theta), R_z(\psi)$$

回転の順序はx軸まわり，y軸まわり，z軸まわりの順になる。三次元では回転の順番がかわると結果が違ってくる。マトリクスの順番には気をつけてほしい。

以上，平行移動と回転を合成すればワールド→ビュー平面の変換ができる。結局，

$$(x_V, y_V, z_V, 1)=M(x_W, y_W, z_W, 1)$$

$$\text{, where } M=R(\varphi, \theta, \psi)T(\Delta x, \Delta y, \Delta z)$$

となる。R(φ, θ, ψ)は具体的には，

$$R(\varphi, \theta, \psi)=R(\varphi)R(\theta)R(\psi)$$

$$=\begin{pmatrix}\cos\psi\cos\theta & \sin\psi\cos\varphi+\sin\psi\sin\theta\sin\varphi & \sin\psi\sin\varphi-\cos\psi\sin\theta\cos\varphi & 0 \\ -\sin\psi\cos\theta & \cos\psi\cos\varphi-\sin\psi\sin\theta\sin\varphi & \cos\psi\sin\varphi+\sin\psi\sin\theta\cos\varphi & 0 \\ \sin\theta & -\cos\theta\sin\varphi & \cos\theta\cos\varphi & 0 \\ 0 & 0 & 0 & 1\end{pmatrix}$$

となる。実際にプログラミングするときは平行移動は加算で行う。回転は各点の座標データに上記のR(φ, θ, ψ)を掛けなければならないので，非常に時間がかかる。そのためこの部分をハードウェア化したグラフィックシステムも多いようだ。

2.クリッピング
(Cohen-Sutherlandのアルゴリズム)

私たちが物を見るとき，すべてが見えるわけではない。視野内にある物しか見ることはできない。今，視野を上下45°，左右45°としてみよう。すると見えるのは**図4**の四角すいの領域になる。

これを視野ピラミッドと呼ぶ。すると，視野と直線には次の4つの場合が起きる。

1．両端がピラミッド内(**図4・a**)
2．片方が内，片方が外(**図4・b**)
3．両端が外だが交わる部分がある(**図4・c**)
4．両端が外で交わらない(**図4・d**)

2や3のようにピラミッドと交わるときは，交点を求める必要がある。このように直線から視野ピラミッド内の部分を切り出すことをクリッピングという。クリッピングにはいくつかの方法があるが，もっとも有名で高速なアルゴリズムに，Cohen-Sutherlandのアルゴリズムがある。これは両方とも内側か外側で，かつ交わらない直線，つまりクリッピングの必要のない直線をいちはやく決定することができる。したがって「大半の直線が視野内」とか「大半の直線が視野外」のときに威力を発揮する。具体的に解説しよう。

直線P_1P_2，($P_1(x_1, y_1, z_1)P_2(x_2, y_2, z_2)$)をクリッピングすることにする。視野ピラミッドとビュー平面座標系を**図5**のように設定する。すると視野ピラミッドを構成する面はそれぞれ，

$y=x$，$z=x$，$y=-x$，$z=-x$

となる。ここで次のような4ビットのフラグを考えよう。

ビット0……$y>x$ならば1
ビット1……$z>x$ならば1
ビット2……$y<-x$ならば1

図5

図4

ビット３……$z<-x$ならば１

視野ピラミッドをx軸に垂直な平面で切ると次のようになる。

これと今のグラフとの関係は，

0110	0010	0011
0100	0000	0001
1100	1000	1001

点P_1のフラグをFLAG1，点P_2のフラグをFLAG２とすると，

１．(FLAG１ AND FLAG２)≠０　ならば交わらない

２．(FLAG１＝０)AND(FLAG２＝０)ならば２点とも内側

３．(FLAG１≠０)AND(FLAG２＝０)ならばP_1が外，P_2が内

４．(FLAG１＝０)AND(FLAG２≠０)ならばP_1が内，P_2が外

５．(FLAG１≠０)AND(FLAG２≠０)ならば交わるかもしれないし交わらないかもしれない

と結論づけられる。１，２の場合は交点を求める処理は不用だ。問題は３，４，５の場合である。

３，４，５はまったく同じ方法で処理できる。まずFLAG１を調べて，FLAG1≠０ならばFLAG＝FLAG１に，そうでなければFLAG＝FLAG２にする。

FLAGのビット０が１ならばABと$y=x$の交点を，ビット１が１ならばABと$z=x$の交点を，ビット２が１ならばABと$y=-x$の交点を，ビット３が１ならばABと$z=-x$の交点を求める。

そしてその交点をCとしてFLAG1≠０ならば直線ACを，そうでなければ直線BCをクリッピングする。直線が先述の１の「交わらない」か，２の「２点とも内側」かのどちらかになるまでくり返せばよい（**図６**参照)。

直線と$y=\pm x$，$z=\pm x$との交点の計算，たとえば２点(x_1, y_1, z_1)，(x_2, y_2, z_2)を通る直線の方程式は，

$$\frac{x-x_1}{x_2-x_1}=\frac{y-y_1}{y_2-y_1}=\frac{z-z_1}{z_2-z_1}=t$$

図６

ゆえに，

$$\begin{cases} x=x_1+(x_2-x_1)t \\ y=y_1+(y_2-y_1)t \\ z=z_1+(z_2-z_1)t \end{cases}$$

となる。一方，平面が$y=x$のときは，

$$y_1+(y_2-y_1)t=x_1+(x_2-x_1)t$$

より，

$$t=\frac{y_1-x_1}{(x_2-x_1)-(y_2-y_1)}$$

よって交点は，

$$\left(x_1+\frac{(x_2-x_1)(y_1-x_1)}{(x_2-x_1)-(y_2-y_1)},\quad y_1+\frac{(y_2-y_1)(y_1-x_1)}{(x_2-x_1)-(y_2-y_1)},\quad z_1+\frac{(z_2-z_1)(y_1-x_1)}{(x_2-x_1)-(y_2-y_1)}\right)$$

となる（ほかの平面もまったく同様)。

この交点を求める部分は乗，除算が必要なため，アセンブリ言語でこのアルゴリズムを使って高速処理をさせるような場合，CPUは乗除算命令を持ったものを使うことが望ましい。８ビットCPUのように乗除算のないCPU上で動く三次元パッケージでは，交点を求めるときに今のように乗除算を利用するのではなく，「中点再分割（midpoint subdivision)」という方法を使うこともある。中点再分割法はCPUでやるため，というよりクリッピングディバイダというハードロジックで組むために作られたアルゴリズムだ。

図７のような場合，ABの中点Mを求める

図７

には，

$$M=\left(\frac{x_1+x_2}{2},\ \frac{y_1+y_2}{2},\ \frac{z_1+z_2}{2}\right)$$

のように加算と右シフトですむ。Mに対して先述のFLAGによるin outチェックをすれば，線分MBは２点とも内側で交点を求める必要がなく，あとは線分AMに対して同処理をくり返せばよい。

CDの場合もまず，線分MDは外側で何の処理も必要なく，線分CMに対して処理をくり返せばよい。

このアルゴリズムの利点はシフト，加算，AND，コンペアで実現できる点にある。今回のプログラムは中点再分割ではなく，乗除算で行った。

Cohen-Sutherlandのクリッピングアルゴリズムより速いアルゴリズム*もある。これらはいずれも，線分と端点に対するチェックを多く行って，不要な交点計算の時間を除いたものだ。

＊参考文献

- Fryer,R."A Fortran windowing Technique for Simulation and CAD," Proceeding Vector General User's Group, 1972
- Jarvis,J.F., "Two Simple Windowing Algorithms," Software Practice and Experience, 5 ,1975

3. 透視変換

私たちが物を見る場合，遠くの物ほど距離に反比例して小さく見える。この計算は一点透視の場合きわめて簡単で，

$$x_d=y_v/x_v,\qquad y_d=z_v/x_v$$

と，単にx座標で割ればよい。今，±45°の視野ピラミッドでクリッピングしてあるので，$-1\leqq x_d\leqq 1$，$-1\leqq y_d\leqq 1$である。デバイス座標系（CRTの座標系）はFM-11では**図８**のようになっている。視野が上下，左右と

も±45°なので画面の400×400の部分を使うとすれば，ビュー平面座標系からデバイス座標系に変換するには，

$$\begin{cases} x_d=200+198*y_V/x_V \\ y_d=200-198*z_V/x_V \end{cases}$$

となる。

FM-7，FM-8では画面は640×200で，しかも縦，横のドット幅の比が2：1なのでその補正も必要になってくる。この場合，

$$\begin{cases} x_d=200+198*y_V/x_V \\ y_d=100-\ 98*z_V/x_V \end{cases}$$

となる。

以上は一点透視の式である。投影ではよく，一点透視，二点透視，三点透視という言葉がでてくる。**図9**を見てもらえば理解できると思うが，投影方向と投影面が直交するのが一点透視，投影面がz軸まわりに回転しているのが二点透視，さらにy軸まわりにも回転しているのが三点透視だ。

*　　　*

以上のことをプログラムにした。ワールド座標系は左手系になっている。データのフォーマットはたとえば，P_1（100，100，100），P_2（200，200，200），P_3（300，300，300）で白の三角形$P_1P_2P_3$を描かせたいとき，10000行から

DATA　100，100，100
DATA　200，200，200
DATA　300，300，300
DATA　"end of dot data"
DATA　1，2，7（1番目と2番目の点をカラーコード7で結ぶ）
DATA　2，3，7
DATA　3，1，7
DATA　"end of connection data"

と入力する。そして，見る位置，見る方向を質問してくるので，ワールド座標系のどの位置から見るか，どの方向を見るかをあたえる（方向はラジアン）。z軸まわりの角度をヘッディング，y軸まわりをピッチ，x軸まわりをバンクという。

データがないと物足りないので，フライトシミュレータで使った地形，山，塔のデータをつけておいた。RUNしたとき，XCG，YCG，ZCGに−10000，0，2000，HEADING，PITCH，BANKに0，0，0と入れて，碁盤目状の地形，山，塔が出ればOKだ。ワイヤフレームによる三次元出力ならば，これくらいの短いプログラムで可能である。実用的な三次元パッケージでは，プログラム本体よりもいかにデータを入れやすくするか，また入力データをいかにして容易に変更できるようにするか，などに重点がおかれている，と言える（三次元タートルグラフィックを使ったり，基本データで物体を作ったりなど）。

大型のグラフィックシステムでも，三次元データの入力は完成されているとはいえず，会話型データ入力はこれからの課題の一つといえるだろう。

FM-11用リスト（FM-7，8では緑色の個所を直してください）


```
1000 '****************************
1010 '*                          *
1020 '* WIRE FRAME DEMONSTRATION *
1030 '*                          *
1040 '****************************
1050 '
1060 GOSUB 2000 :' INTIALIZE ROUTINE
1070 GOSUB 3000 :' INPUT VEIWING DATA
1080 GOSUB 4000 :' CO-ORDINATES TRANSFORMATION
1090 GOSUB 5000 :' CLIPING, PERSPECTIVE PROJECTION AND DRAWING
1100 END        :' END OF MAIN ROUTINE
2000 '*
2010 '* INTIALIZE ROUTINE
2020 '*
2030 MAXDOTNUMBER=1000:DIM X(MAXDOTNUMBER),Y(MAXDOTNUMBER),Z(MAXDOTNUMBER)
2040 SCREEN 5:CLS:LINE(0,0)-(399,399),PSET,7,B
2050 RETURN : ' end of intialize routine
2060 '
3000 '*
3010 '* VEIWING DATA INPUT ROUTINE
3020 '*
3030 INPUT "XCG,YCG,ZCG ";XCG,YCG,ZCG
3040 INPUT "HEADING,PITCH,BANK ";HEADING,PITCH,BANK
3050 RETURN : ' end of viewing data input routine
3060 '
4000 '*
4010 '* COORDINATES TRANSFORMATION ROUTINE
```

（緑色注記：2040行の399 → 199；SCREEN 5: → FM-7はSCREEN 7，7　FM-8は削除）

```
4020 '*
4030 GOSUB 4200 :' SET COORDINATES TRANSFORMATION MATRIX ELEMENTS
4035 I=1
4040 READ FINP$:IF FINP$="END OF DOT DATA" THEN RETURN
4050 X1=VAL(FINP$) : READ Y1,Z1 : GOSUB 4400 :' MUL. MATRIX
4060 X(I)=X2:Y(I)=Y2:Z(I)=Z2 : I=I+1 : GOTO 4040
4190 '
4200 ' SET COORDINATES TRANSFORMATION MATRIX
4205 '
4210 CH=COS(HEADING) : CP=COS(PITCH) : CB=COS(BANK)
4220 SH=SIN(HEADING) : SP=SIN(PITCH) : SB=SIN(BANK)
4230 '
4240 A11=CH*CP  : A12=SH*CB+SH*SP*SB : A13=SH*SB-CH*SP*CB
4280 A21=-SH*CP : A22=CH*CB-SH*SP*SB : A23=CH*SB+SH*SP*CB
4290 A31=SP     : A32=-CP*SB         : A33=CP*CB
4310 '
4360 RETURN
4400 '
4410 ' MUL. MATRIX  (X2,Y2,Z2)=M(X1,Y1,Z1)
4420 '
4430 X=X1-XCG : Y=Y1-YCG : Z=Z1-ZCG
4440 X2=A11*X+A12*Y+A13*Z : Y2=A21*X+A22*Y+A23*Z : Z2=A31*X+A32*Y+A33*Z
4450 RETURN
4455 '
4460 ' END OF COORDINATES TRANSFORMATION ROUTINE
4470 '
5000 '*
5010 '* CLIPING, PERSPECTIVE PROJECTION AND DRAWING ROUTINE
5020 '*
5030 READ FINP$ : IF FINP$="END OF CONNECTION DATA" THEN RETURN
5040 I=VAL(FINP$) : READ J,COLOUR
5050 X1=X(I) : Y1=Y(I) : Z1=Z(I) : X2=X(J) : Y2=Y(J) : Z2=Z(J)
5060 GOSUB 6000 :' cliping
5065 IF CONDITION$="not intersect" THEN 5030
5070 GOSUB 5600 :' perspective projection
5080 LINE(XX1,YY1)-(XX2,YY2),PSET,COLOUR
5090 GOTO 5030
5460 '
5600 ' perspective projection
5610 IF X1=0 THEN X1=1
5620 IF X2=0 THEN X2=1
                                  100          98
5630 XX1=Y1/X1*198+200 : YY1=200-Z1/X1*198
5640 XX2=Y2/X2*198+200 : YY2=200-Z2/X2*198
                                  100          98
5650 RETURN
5660 '
5670 ' end of clipping perspective projection drawing routine
5680 '
6000 ' clipping
6010 X=X1:Y=Y1:Z=Z1:GOSUB 6140:FLAG1=FLAG
6020 X=X2:Y=Y2:Z=Z2:GOSUB 6140:FLAG2=FLAG
6030 '
6040 IF (FLAG1 AND FLAG2)<>0 THEN CONDITION$="not intersect":RETURN
6050 IF (FLAG1=0) AND (FLAG2=0) THEN CONDITION$="do intersect":RETURN
6060 IF FLAG1<>0 THEN FLAG=FLAG1 ELSE FLAG=FLAG2
6070 DX=(X2-X1):DY=(Y2-Y1):DZ=(Z2-Z1)
6080 IF FLAG AND 1 THEN T=(X1-Y1)/(DY-DX):Y=Y1+DY*T:X=Y:Z=Z1+DZ*T:GOTO 6120
6090 IF FLAG AND 2 THEN T=(X1-Z1)/(DZ-DX):Z=Z1+DZ*T:X=Z:Y=Y1+DY*T:GOTO 6120
6100 IF FLAG AND 4 THEN T=-(X1+Y1)/(DX+DY):Y=Y1+DY*T:X=-Y:Z=Z1+DZ*T:GOTO 6120
6110 IF FLAG AND 8 THEN T=-(X1+Z1)/(DX+DZ):Z=Z1+DZ*T:X=-Z:Y=Y1+DY*T
6120 GOSUB 6140
6130 IF FLAG1 THEN FLAG1=FLAG:X1=X:Y1=Y:Z1=Z:GOTO 6040 ELSE FLAG2=FLAG:X2=X:Y2=Y
:Z2=Z:GOTO 6040
6140 FLAG=0
6150 IF Y>X THEN FLAG=1
6152 IF Z>X THEN FLAG=FLAG OR 2
6155 IF Y<-X THEN FLAG=FLAG OR 4
6165 IF Z<-X THEN FLAG=FLAG OR 8
6170 RETURN
10000 ' earth
10010 DATA 10000,10000,0
10020 DATA 10000, 9000,0
```

```
10030 DATA 10000, 8000,0
10040 DATA 10000, 7000,0
10050 DATA 10000, 6000,0
10060 DATA 10000, 5000,0
10070 DATA 10000, 4000,0
10080 DATA 10000, 3000,0
10090 DATA 10000, 2000,0
10100 DATA 10000, 1000,0
10110 DATA 10000,    0,0
10120 DATA 10000,-1000,0
10130 DATA 10000,-2000,0
10140 DATA 10000,-3000,0
10150 DATA 10000,-4000,0
10160 DATA 10000,-5000,0
10170 DATA 10000,-6000,0
10180 DATA 10000,-7000,0
10190 DATA 10000,-8000,0
10200 DATA 10000,-9000,0
10210 DATA 10000,-10000,0
10220 DATA -10000,10000,0
10230 DATA -10000, 9000,0
10240 DATA -10000, 8000,0
10250 DATA -10000, 7000,0
10260 DATA -10000, 6000,0
10270 DATA -10000, 5000,0
10280 DATA -10000, 4000,0
10290 DATA -10000, 3000,0
10300 DATA -10000, 2000,0
10310 DATA -10000, 1000,0
10320 DATA -10000,    0,0
10330 DATA -10000,-1000,0
10340 DATA -10000,-2000,0
10350 DATA -10000,-3000,0
10360 DATA -10000,-4000,0
10370 DATA -10000,-5000,0
10380 DATA -10000,-6000,0
10390 DATA -10000,-7000,0
10400 DATA -10000,-8000,0
10410 DATA -10000,-9000,0
10420 DATA -10000,-10000,0
10430 DATA  9000,10000,0
10440 DATA  8000,10000,0
10450 DATA  7000,10000,0
10460 DATA  6000,10000,0
10470 DATA  5000,10000,0
10480 DATA  4000,10000,0
10490 DATA  3000,10000,0
10500 DATA  2000,10000,0
10510 DATA  1000,10000,0
10520 DATA     0,10000,0
10530 DATA -1000,10000,0
10540 DATA -2000,10000,0
10550 DATA -3000,10000,0
10560 DATA -4000,10000,0
10570 DATA -5000,10000,0
10580 DATA -6000,10000,0
10590 DATA -7000,10000,0
10600 DATA -8000,10000,0
10610 DATA -9000,10000,0
10620 DATA  9000,-10000,0
10630 DATA  8000,-10000,0
10640 DATA  7000,-10000,0
10650 DATA  6000,-10000,0
10660 DATA  5000,-10000,0
10670 DATA  4000,-10000,0
10680 DATA  3000,-10000,0
10690 DATA  2000,-10000,0
10700 DATA  1000,-10000,0
10710 DATA     0,-10000,0
10720 DATA -1000,-10000,0
10730 DATA -2000,-10000,0
```

XCG : 0, YCG : 0, ZCG : 13000
HEADING : 0, PITCH : 1.571, BANK : 0

XCG : −10000, YCG : 0, ZCG : 5000
HEADING : 0, PITCH : 0.8, BANK : 0

XCG : −10000, YCG : −15000, ZCG : 4000
HEADING : 1, PITCH : 0.2, BANK : −0.2

XCG : −10000, YCG : 5000, ZCG : 500
HEADING : −0.5, PITCH : 0, BANK : 0

```
10740 DATA -3000,-10000,0
10750 DATA -4000,-10000,0
10760 DATA -5000,-10000,0
10770 DATA -6000,-10000,0
10780 DATA -7000,-10000,0
10790 DATA -8000,-10000,0
10800 DATA -9000,-10000,0
10810 ' mountain
10820 DATA 10000,-6000,1000
10830 DATA 10000,-5000,500
10840 DATA 10000,-3500,800
10850 DATA 10000,-2000,600
10860 DATA 10000,0,1000
10870 DATA 10000,2000,400
10880 DATA 10000,4000,600
10890 '
10900 ' tower
10910 DATA 0,   00,0
10920 DATA 1000,  000,0
10930 DATA 1000,-1000,0
10940 DATA 0,-1000,0
10950 DATA 500,-500,1000
10960 DATA 'END OF DOT DATA'
10970 DATA 1,22,7,2,23,7,3,24,7,4,25,7,5,26,7,6,27,7,7,28,7,8,29,7,9,30,7,10,31,
7,11,32,7,12,33,7,13,34,7,14,35,7,15,36,7,16,37,7,17,38,7,18,39,7,19,40,7,20,41,
7,21,42,7
10980 DATA 1,21,7,43,62,7,44,63,7,45,64,7,46,65,7,47,66,7,48,67,7,49,68,7,50,69,
7,51,70,7,52,71,7,53,72,7,54,73,7,55,74,7,56,75,7,57,76,7,58,77,7,59,78,7,60,79,
7,61,80,7,22,42,7
10990 ' mountain
11000 DATA 21,81,4,81,82,4,82,83,4,83,84,4,84,85,4,85,86,4,86,87,4,87,1,4
11010 ' tower
11020 DATA 88,92,2,89,92,2,90,92,2,91,92,2
11030 DATA 'END OF CONNECTION DATA'
```

コンピュータについての軽いお話

林 晴比古

私の友人に，コンピュータのことを，——あれは本当は電気計算機と呼ぶべきじゃないかね，という男がいます。その理由は，——コンピュータは電源コードをコンセントに差し込んで動作している。あれは決して電子ではない。また内部の計算も0ボルトと5ボルトの2つの電気信号でやっているのだから，もはや電気の主導権は動かしがたい，というわけです。

そう言われると，近頃はやりのマイコン制御付きであっても，電子釜とか電子冷蔵庫とは言わない。なんだか「子」を使うか「気」を使うかは，歴史的な経過あるいは多分にムード的なものが働いているような気がします。

このように少しは専門家に近い立場の者であっても，電子計算機というものを，わりと曖昧にとらえているところがあります。

ましてや初心者にあっては，その定義を的確に述べることなど，とてもほど遠いことでしょう。

——コンピュータとは，つまり，なんというか，その，まあ，たとえば，一口に言って，早い話，あれだね。これでは，まさしくカオスの概念そのものです。

そこでコンピュータというものについて，多少は知識になりそうなことを，軽くお話してみたいと思います。

テレビとコンピュータ

テレビはなぜ映るのでしょう？　という問題を出されたとします。あなたはどう答えますか？

放送局からの電波を受信し，チューナにて選局し，走査線を……，これはマル。

会社の方針だから……，これはバッテン。

二重丸の答えは，「スイッチをひねるから」です。

これは別に茶化しているわけではなく，コンピュータ，とくにソフトウェアの世界ではたいへん重要な発想です。なぜならソフトウェア部門においては創造性こそが一番大切なものであり，たとえばある一つの命令について，なぜそうなるのだろうと追究することは，なんら生産的な行為とならないのです。

動作原理はよくわからないけれど，とにかくこれだけの命令があるのだから，うまく組み合わせたら，おもしろいことができるのではないかしら，と頭をひねることこそが，ソフトウェアの世界の知性であり，全てなのです。

実際，私たちの大部分はテレビの原理など知りはしないのに，生活必需品として充分に使いこなしています。年端もいかぬ幼児が，チャンネル権を握っていたりします。

——コンピュータはテレビである。

まずは大胆にそういうとらえ方をしてみましょう。

ひとことでいうと

とはいっても，まさかコンピュータを指さして，

「テレビがある」

では，国定忠次になってしまいます。では，簡単に言って，コンピュータとはどういうものなのでしょう。

——計算機と名がついているのだから，電卓の親分である。

以前まではこの答えでもそれほど的外れではありませんでしたが，最近はこれでは少し不充分のようです。

——コンピュータとは情報を処理する機械である。

これが現在のところ，一番ピッタリくる解答でしょう。もっと詳しく言うと，入手可能な情報（入力）を，上手に加工して（処理），望みの情報（出力）を作成するものが，コンピュータと呼ばれる機械の能力なのです（**図1**参照）。

この時，情報というのは数字であったり，物の名前であったり，文章であったりする

わけで，たとえば

1＋2＝3

の計算を行わせるとすると，入力は1と2，処理はプラス，出力は3に相当するわけです。

同じくAさん（男），Bさん（女），Cさん（男）の中から，女性だけを抽出する場合は，A，B，Cさんが入力，「女性だけを抽出」が処理，Bさんが出力となります。

これらの単純な仕事の繰り返しで，コンピュータは最終的には大きな仕事をやりとげるのです。

ところで，以前，テレビによく出る評論家が，盗作問題で糾弾されたとき，詳しい一字一句は忘れましたが，

——活字になっている情報を適当に寄せ集めて一冊の本を作ることも，立派な創作である。

というようなことをしゃべっていました。

それを自分のオリジナルとして出版することの是非はともかくとして，これはまさしく情報処理そのものではないかと，私は

思ったものです。

それにしても，手帳一冊で本ができるとは，思えませんが……。

コンピュータは間違えない？

よく銀行の窓口嬢や，電電公社の職員が，お客に，

「コンピュータで計算していますので，間違いありません」

と言ってひんしゅくをかいます。

これに対して，またもや巷の評論家が，

「コンピュータでも間違いはやる」

としたり顔で述べます。

どちらが正しいと思いますか？

正解は窓口嬢の方です。

コンピュータは決して間違いはしないのです。しかし，「決して間違えない」という，そのこと自体が実はトラブルの原因になっているのです。

たとえばあるサラリーマンが，会社の命令で，薄給をむりやり銀行振り込みにさせられたとします。薄給なので当然彼は，給料日の翌日に，ほとんど全額を引きおろしてしまいます。

そのような空っ風の口座に，ある日突然，1,000万振り込まれたらどうでしょう。もし昔ながらに人間が作業しているなら，直感で，

――これは何か変だぞ！

と警戒心を持つはずです。

しかしコンピュータなら，数字上のつじつまが合うかぎり，そのまま無条件に振り込んでしまうのです。

つまりコンピュータは決して間違いはしないのですが，そのプログラムを組んだり，入力操作をするのが完全無欠とは言えない人間であるために，トータルなコンピュータシステムというものは，誤りを犯すのです。

ですからコンピュータシステムが，信じられないほどの仕事をやっていても，コンピュータ自身は今自分が何をやっているのか，理解しているわけではありません。コンピュータは人間のセットしたプログラムを１ステップごとに忠実に，しかも非常に高速に処理しているだけなのです。

コンピュータをときおり，「偉大なる馬鹿」と呼ぶのは，そういう意味では，とても的確な表現だと思います。ここまで述べると，

――コンピュータは何でもできる。

というのは誤りだとわかるでしょう。

少なくとも「コンピュータで」というべきです。

原理

コンピュータの基本機能は，

入力：キーボードなど

出力：プリンタなど

記憶：主記憶装置やフロッピー

演算：レジスタや加算器

制御：命令の実行管理

の５つに分けられます。

制御機能はピンとこないかもしれませんが，命令を１度に１個ずつ，順序正しく行わせる役目を持っています。

これらの詳細な説明は，この稿の目的ではありませんので他の機会に譲るとして，ここでは演算法について少し述べてみます。

天文や気象の計算を行うCRAY-1などのスーパーコンピュータをはじめとして，電子計算機はその名のとおり，数値計算が最も得意です。さぞかし高度な演算機能が内蔵されているかのように思えます。

しかし，非常に大胆な言い方をすると，実はコンピュータはたった１種類の計算しかできないのです。

そのたった一つの機能とは「たし算」です。

たし算を実現する回路は，加算器あるいはアダーと呼ばれていますが，いわば専門家だけが操作する回路であるため，8080などのLSI化されたプロセッサレベルになると，もうユーザーズマニュアルにもほとんど出てこない用語になります。

ところで，なぜたし算だけで，高度な計算ができるのかということになりますが，それは次のようなことで解決されているのです。

まずSIN, COS などの三角関数や，微分，積分などのあらゆる数学的計算は，四則演算により求められます。

逆に言えば，四則演算さえできれば，どのような計算も実現できることになります。そしてその四則計算はたし算だけで可能なのです。具体的に述べてみましょう。

①たし算

これはそのまま加算によりできます。

②ひき算

たとえば，

20－４

を求める場合，４という数字を２進数の補数形式という特別な形に変えて，それを加算します。わかりやすい概念で示すと

20－４＝20＋(－４)

というように説明できます。

③かけ算

これはたし算の繰り返しで実行します。

たとえば，20×４は，

20＋20＋20＋20

と，20を４回加算して求めます。

④割り算

これは引き算の繰り返しです。

20÷４＝20－４－４－４－４－４

と，４の引き算可能回数を数えて，商の５を求めます。もちろん，このときの引き算も，②の例のように実はたし算にして，実行されます。そのほか，比較や，シフトも加算器にて実現できます。アダーは計算機の頭脳そのものなのです。

ハードウェア

ソフトウェアの対立概念として，計算機本体や，磁気テープ装置，プリンタなど，実体のあるものを総称して，ハードウェアと呼びます。しばしば「金物」と日本語に訳されますが，あまりピッタリしたイメージではないような気がします。

システムの構成は大別すると，

①入出力装置

②メモリ

③演算装置

に分けられます。

入出力装置は，文字どおりデータを入出力するもので，最近は非常に高級な装置が開発されています。

たとえば入力するのに，いちいちキーインしなくてもよい，文字読み取り装置，音声認識装置，手書き文字入力装置などです。

出力装置の代表はなんといってもプリンタです。これも初期のものはTTYより，多少ましな程度でしたが，今では１秒間に300行以上もプリントするものもあります。

またレーザービームプリンタという，レーザー光線にて感光部の電荷をコントロールしてプリントするシステムは，１㎜の幅に10個のドット（点）を打てる分解能を持っています。

一方，メモリは技術革新の最先端を走っている素子です。64K RAMと呼ばれるものは，1個のＬＳＩで65,536（$=64\times2^{10}$）ビットもの情報を持つことができます。

つい7～8年前は，コアメモリが主流で，背よりも高い鉄の架にわずか8Kワード(×32ビット）という主記憶装置もありましたが，今ではそれが基板1枚になってしまいました。その頃からの生き証人の一人として，はなはだ感慨深いものがあります。

ところでメモリには，今述べた主記憶というコンピュータ本体と一体になったメモリのほかに補助記憶があります。それは磁気テープや，フロッピーディスクなどで，主記憶に入りきらないデータを一時的に保持したり蓄積したりするものです。

ちょうど，財布には必要なお金だけを入れて持ち歩き，財産は銀行に預けておくのと同じようなことです。

ビットの話

順序が逆になりましたが，コンピュータは，ビット単位にて動作しますので，これについて少し，述べておきましょう。

BitとはBinary Disitの略で，オンとオフの2つの状態で現される情報の単位です。

部屋の電灯のスイッチをオン／オフすれば，電灯がついたり消えたりと，2つの状態になりますので，これも1ビットと表現することができます。

また，「赤あげて，白あげて……」と旗を上下させるのも1ビットと言えます。

このビット単位で数値を表現する方法を2進法と言います。これに対して私たちの日常の数字は10進数であり，時計は60進数です。

ところで，改めて，10進数とはどういう数字かといいますと，1の位が9より大きくなり，もう表示できる記号がなくなったら，1つ桁上がりする数値表現法のことです。これと同様に，2進数とは1の位が1より大きくなったら，桁上がりする方法です。

図2　10進数と2進数の比較

10進数	2進数
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

ではなぜコンピュータは2進数を用いているのでしょう。それは2進数が，コンピュータにとって都合のいい方法であるためなのです。いわばコンピュータの勝手な都合により，われわれ人間にとっては，なじみにくい2進数が採用されているのです。

その都合とは，たとえば，

①論理計算が容易

②記憶しやすい

③信頼性が高い

などです。

もう一つ，なるほど2進数の方が有利だという例をお見せしましょう。

図2における0～7の記号，8種をランプにて表示させることを考えます。まず10進数の方は，**図3**のように8個のランプに

図3

対応して0～7の数字をマジックで書くなどして，5のランプが点灯したら信号は5というように判定するでしょう。

これに比べて，2進数では，**図4**のように，ランプのそばに，

図4

4，2，1

の3つの数字を書いておきます。(これをウエイト＝重みという）。そしてランプが点灯しているところに書いてある数字のみを加えることにします。たとえば**図2**の数値5の場合，2進数では両側が1になっています。そこで**図4**でも両側のランプのみ点灯させ，そこに書いてある数字4と1をたし算して，5の信号であると知るわけです。

このように同じ8種類の信号を伝えるとき，10進数では8個のランプが必要なのに対して，2進数ではわずか3個で済むのです。

もちろん10進数でも，ランプの光量を0～9の10段階にすれば1個のランプで事足りるわけですが，それでは判定があやふやになります。この点，2進数ならオン・オフの二者択一ですから，コンピュータにとって非常に判別しやすく信頼性の高いものになります。

人間の指は5本ずつの10本だから10進数が主流になる。もし4本か6本だったら……？

2進数の有利さが多少はおわかりいただけたことと思います。

ところで初心者の中には2進数の感覚が，どうもピンとこないという人がいます。

そういう人のために特別に2進数マスターの秘伝をお教えしましょう。

だれでも，ソロバンの動かし方ぐらい知っていると思いますが，あの手動計算器の数値は10進数と5進数で表現されています。

なぜ5進数かというと，下部にある4個の玉がそれぞれ1を表すのに対し，上部にある玉だけは1個で5の数値を表すからです。

そこでまず下部にある4つの玉を，動かないようにヒモででも縛ってしまいます。次に上部にある1個の玉を数値5でなく，1だと考えます。そのまま1，2，3，4，と数を入れていってみて下さい。実にスムーズに2進数を体験されることでしょう。

私自身このマスター法を知ったとき，一瞬にして2進数を体得してしまい，「教育とは一体何なのだろう」と，まじめに考えこんでしまったことがあります。

2進数を16進数に変換するには下位から4ビットずつ区切って，8，4，2，1の重みをたしていくと簡単です。

ソフトウェアの重要性

「コンピュータ，ソフトがなければ，た

だの箱」

とよく言われます。箱ならまだオモチャでも入れることができますが，この場合はもっとひどい，単なる鉄クズです。

コンピュータはソフトで動くということはもう，先刻ご存知のことでしょう。ミニコン以上になると，コンソールと呼ばれる操作盤よりのキー指示（マシン語の入力などを行う）などは，論理回路で構成されたりしますが，パソコンレベルでは，電源を入れた瞬間から，全てがソフトウェアにより実行されます。

したがって，その基本になるソフトウェアの良し悪しが，コンピュータのトータルパフォーマンスを決定する重要事項になります。

たとえばA(10万円)，B(20万円)の2つのパソコンがあったとします。

そのマシン語モニタの開発担当者として，Aの側に非常に優秀なシステムエンジニアが採用されたとします。ここでいう優秀なというのは，単に良いプログラムを設計するというのではなく，人間工学や，心理学，あるいは美的センスに富んでいるということも含みます。

理想的なモニタができたとすると，

①コマンドの覚えやすさ
②オペランド表現の適切さ
③省略時解釈の妥当性
④体系的機能（似たような機能がなく，足りない機能がない）
⑤コマンドの簡潔さ
⑥使用時の安定感
⑦エラー回復処理および情報の適切さ

などの点で，大変使いやすいモニタになります。私はよく，以上のことをひとまとめにして，

——コマンドが分らなくなったとき，多分こうだろうと適当にキーインして，それが当たる確率の高いほど，よくできたソフトウェアである。

と説明します。

このように，優れたモニタの付いているAというパソコンが，Bという20万円のパソコンより，コストパフォーマンスが高いという例は，コンピュータの世界ではしばしばあります。

そのようなコンピュータの性能を左右するソフトウェアの代表格は，なんといってもOS(オペレーティングシステム)でしょう。パーソナルコンピュータ用のOSとして最も有名なものに，CP/Mがあります。CP/Mはパーソナルコンピュータの持っている能力を最高に引き出すためのソフトウェアです。たとえつまらない，どこといって特徴のないパソコンであっても，CP/Mを採用すると，途端に世界に通用するコンピュータになるのです。

現在，新しくコンピュータシステムを客先の注文により設計すると，全コストのうち，ソフト開発費の占める割合が50%を越えるのは，もはや常識となっています。

パソコンに限ってみても，本体価格が，たかだか10〜20万円なのに対し，Pascalや，COBOLなどのソフトウェアはフロッピー1枚で20〜30万円もします。もし自分でビジネスプログラムを作ろうとした場合でも，能力は当然あるものとして，本格的なものは，すぐに1か月くらいかかってしまいます。そうすると1日8時間で25日かかるとして，時給を千円とすると，8×25×1,000＝20万円のコストがかかることになります。価格だけをとってみても，ソフトウェアの方がとても重要なのはおわかりいただけると思います。

次にソフトウェアの重要性には，もう一つ別の意味があります。

それは，ある特定のコンピュータで走らすことのできるソフトウェアが，どれだけ流通しているかということです。流通しているソフトウェアの数が多いほど，そのマシンを使いこなすには有利となります。

もう新型が出たので言ってもいいと思いますが，NECのPC-8001というパソコンは発売時には，当時としては驚異的な性能と低価格でセンセーショナルな話題を呼びました。しかし1982年には専門家的な見方をすると

——あれはどうも……

という製品になってしまいました。

しかし現実に，PC-8001mkIIが発売になる直前まで，オリジナルの方はソフトの蓄積量の影響で売れ続けたわけです。

これもソフトウェアの重要性を証明する一例です。

BASIC上達法

最後にBASICの上達法を述べて，この稿を終わりにしたいと思います。

皆さんの中には，きっと，

・パソコンの1日スクールに行ったのだが，さっぱりものにならない。
・本を読んでみたけどちっともわからない。
・機械を買ったが，今は使われずにほこりをかぶっている。

などという人がいるでしょう。

しかし，これらのことを，ただ漫然とやっていても，プログラミングのマスターなどできるはずがありません。なぜなら，ものを覚えるうえでの基本的な動機付けが全くなされていないからです。

それには二つあります。

まず第一は

「追いつめられること」

です。

会社員であれば，上司より，

——パソコンをマスターしろ。

という命令が出たり，学生なら，

——家からの仕送りがなくなったので，ゲームを作って学資にしよう。

とか，また，あこがれの女性が

——パソコンぐらいマスターしている男性でないと好きになれないわ。

と言っているのを聞いた，とかいうことです。とくに3番目の理由は強烈でしょう。

次にもう一つの重要な動機付けは

「興味を持つこと」

です。

興味こそが，学問の原動力なのですから，これなくして，テキストなどを読んでも，積極的に知識を吸収できるはずはないでしょう。

では追いつめられることもなく，とくに興味もない者はどうするか，ということになりますが，その答えは決まっています。

パソコンなどおやめなさい，ということです。

パソコンなどできなくたって，会社を首になることはここ当分ありませんので，そういう人達は，もっとほかの有効なことに時間を使ってもらいたいものです。

コンピュータはそれを必要とする者にとっては，大変強力な武器となります。しかし，必要としない者にとっては，なくてもちっとも困らない，そんな程度のものです。

軽〜いパソコンを貴方もどうぞ。

for FM-7,8,11

FMCALC──その基本機能と応用(1)

(株)ソフトマート
岡部　正

1

FMCALCとは何か？

FMCALCは作表用の簡易言語である。

ビジネスや科学技術を含めた人間の生活には，さまざまな数値データが用いられ，それを表の形に整理し，その加工によって新たなデータを生みだす必要性がいろいろな場面で登場する。簡単なものならば，紙と鉛筆と定規があればこと足りる。紙に縦横の罫を引き，必要な文字や数字を記入し，さらに必要に応じて筆算を行い，横を埋めていけばよい。

しかし，データ量が多かったり計算式が複雑だったりすると，このような方法では間に合わず，コンピュータにその仕事をさせたいと考えるのが自然の成りゆきだろう。

だが，たとえばBASICを使って作表を試みたとき，かなり簡単な表でも長々しいプログラムを書く手間は避けられない。たった一つの表を作成するならそれも我慢できようが，同一の様式の表に，次々，異なったデータを代入して複数の表を比較したいようなとき，プログラムは一層複雑にならざるを得ない。

逆に，財務管理や会計管理用ソフトなどによって作られる表は，完全に「押し着せ」の表である。ユーザーがすることは，膨大なデータの入力という退屈かつ面倒な作業だけで，出力帳表のフォーマットをいじる融通性はほとんどない，と言ってもよい。

システムが大型化すると出力帳表の様式が固定されるばかりでなく，データ入力も専門職の手に委ねなければならないので，実際の数値と異なったデータを入れ，シミュレーションをしてみる可能性もずっと狭められてしまう。

＊　＊　＊

以上述べたように，既成の言語やシステムを使った場合のコンピュータによる作表作業には，いろいろな制限が課されている。

その点FMCALCは，純粋に作表用言語として開発されたソフトウェアなので，

①BASICやFLEX，CP/MなどのOSを必要としない。

②POWER ON 操作のみで起動する。

③起動すると作表画面がいきなりディスプレーに表示される。

という特徴を持っている。

作表画面は**図1**のように現れる。

左側の1～21は行番号，下段のA～Eは列名で，この表示画面においてすでに縦21行，横5列の表の枠組ができあがっている。枠組ができあがっているので，ユーザーは所望の個所にデータや文字を，次々入れていくだけでよい。

起動時に表示された画面はFMCALCの表のごく一部であって，列名A～Zの26列(コマンドによってA～Z，a～zの52列まで拡張できる)，行数128の巨大な表の大部分は現在隠れているにすぎな

図1　起動画面

図2

い（**図2**）。画面の移動により，希望の領域を表示させることができる。

2

FMCALCの基本動作

（1）ブロックと座標

列名（A～Z）と行番号（1～128）によって決まる領域をブロックと呼び，FMCALCへの入力単位となる。その列名と行番号が座標であり，たとえばA1，C23，Y105などと表し，ブロックの位置を決定する。

現在の入力対象ブロックは，ブロックカーソルにより示される（起動画面においてA1の位置にある■■■がブロックカーソル）。データなどを入力してリターンキーを押すと，ブロックカーソルのあるブロックにデータが入り表示される。ブロックカーソルの移動はカーソル移動キー（↑↓←→）が利用できるほか，ブロック座標の指定による方法もある。また後述のCURSORコマンドも，効率的なブロックカーソルの移動に有用である。

（2） 式

FMCALCの有力な武器は，四則演算や各種関数を用いた計算が自由に行えることである。

式に用いることのできる演算は以下のものである。

＋　加算
－　減算
＊　乗算
／　除算
＾　ベキ乗

また，FMCALCで利用できる関数は**表1**のとおりである。

式や関数の対象として，直接数値だけでなく，座標も扱える。そのため入力データの多様な処理が可能だ。例を挙げて説明しよう。**図3**を見てほしい。

たとえば，A2～D5の範囲（斜線部分）には数値データ，E列には各行の合計，E6にはE2～E5の平均値，E7にはE2～E5の標準偏差を計算させて入れたとき，入力すべき式は次のようになる。

E2　＋SUM（A2，D2）
E3　＋SUM（A3，D3）
E4　＋SUM（A4，D4）
E5　＋SUM（A5，D5）
E6　＋AVR（E2，E5）
E7　＋AVR（F2，F5）－E6＾2

ただし，

F2　＋E2＾2
F3　＋E3＾2
F4　＋E4＾2
F5　＋E5＾2

すなわち，F列はワークエリア風の扱いとなる。

（3） 罫

表を見やすくするためには，適切な罫を引くことが不可欠であるが，FMCALCではグラフィックモードにより縦横の罫を引くことができる。二重線や隅，交差の処理もできるよう，グラフィック文字は充分に揃っている。

（4）日本語の活用

入力できる文字は英数字に限らず，カタカナや頻繁に使われる

図3

表1　FMCALCの関数

	関数名	目的	書式	精度	備考
一般関数	SIN	正弦を与える	SIN（<引数>）	13	<引数>の単位はラジアン
	COS	余弦を与える	COS（<引数>）	13	<引数>の単位はラジアン
	TAN	正接を与える	TAN（<引数>）	13	<引数>の単位はラジアン
	ATN	逆正接を与える	ATN（<引数>）	13	<引数>の単位はラジアン 数値範囲は－π／2からπ／2まで
	ABS	絶対値を与える	ABS（<引数>）	16	
	EXP	eのべき乗を与える	EXP（<引数>）	15	
	LOG	自然対数を与える	LOG（<引数>）	15	<引数>は正
	SQR	平方根を与える	SQR（<引数>）	—	<引数>は0または正
	INT	整数化を行う	INT（<引数>）	—	<引数>をこえない最大の整数を与える
	SGN	符号を与える	SGN（<引数>）	—	<引数>が正……………1 <引数>が0……………0 <引数>が負……………－1
特殊関数	SUM	加算を行う	SUM（<座標1>，<座標2>）	16	座標1から座標2までの加算
	AVR	平均を求める	AVR（<座標1>，<座標2>）	16	座標1から座標2までの平均

（市，区，町，村，年，月，日，時，分，秒，円，人）も含まれるので，読み取りやすい帳表を作ることができる。システムとオペレータの対話に用いられるメッセージも，カタカナで表示される。

（5） ファイル

オペレータが文字，数値，式を入力するときの単位は一つの座標によって決まるブロックだが，ディスケットへのSAVE（格納），ディスケットからのLOAD（呼び出し）の単位はファイルである。

ファイルは左上のブロックと右下のブロックの座標により決定し，それぞれに名称をつけることができる。もちろんFMCALCの表全体を一つのファイルにすることも可能であるし，全体をいくつかのエリアに分割しそれぞれをファイルとして取り扱うことも可能である。ファイル名も英数字だけでなく，カタカナなどその他の文字が使用できる。

3 FMCALCのコマンド

作表する際は，以下で紹介するコマンドを使用すると作表が簡易化され読みやすくなり，活用範囲も広がる。

各コマンドは定められたキーを定められた順序で入力することにより，所定の機能をただちに実行する。その際オペレータにメッセージを問いかけ，応答により動作するものもある。

FMシリーズには10個のファンクションキーが装備されているがそのおのおのがFMCALCのコマンドに割り当てられている。ファンクションキーを使用すれば，キー操作をさらに短縮することができる。ここではコマンドをいくつかのグループに分けて説明しよう。

（1） 初期化コマンド

INIT

このコマンドにより現在処理中の表の内容はすべてクリアされ，初期画面（FMCALC起動時の表示画面）に戻る。オプションのパラメータを指定すれば，列数をDEFAULTの26から最大52まで増加できる。27番目以降の列の名称として，英小文字aからアルファベット順に使用される。

（2） 編集コマンド

BLANK

このコマンドで，ブロックカーソルのあるブロックの内容（文字，データ，式）が消去される。式の入っていないブロックでは，単にリターンキーを押すことによって内容は消去されるが，式の入ったブロックの内容をクリアするには，このコマンドが有力である。

COPY

パラメータで指定されたブロック，列，または行の内容を，やはりパラメータで指定されたブロック，列，行にコピーするコマンドである。コピーの対象に式を含めるか含めないかはオプションの指定により可能である。

式を含めてコピーするとき，たとえばコピー元をE列，コピー先をF列として，E10＝＋SUM（E2，E8）ならばコピーの結果，F10＝＋SUM（F2，F8）となる。つまり，システムが自動的に式の座標変換を行ったわけである。

この機能は規則性を持つ表の作成を容易にし，かつ入力エラーの減少に役立つ。システムによる自動座標変換を抑制し，コピー元にある式の座標をそのまま残したいときは，目的の座標に特定文字（@）を付けることにより可能である。

CURSOR

他の多くのコマンドと違いCURSORコマンドの機能は，もう一度CURSORコマンドが入力されるまで持続する。

このコマンドはブロック入力時のリターンキーを押した後，ブロックカーソルの移動方向と移動範囲を指定する。システムの起動にはリターンキーを押してもブロックカーソルは移動しない。文字やデータを一定の範囲にある連続したブロックに，次々入力するとき，このコマンドによりブロック移動キーを不要にする。そのため，操作の能率がアップする。

図4・ⓘのような順序で入力したいときは，CURSORコマンドのパラメータ（実際にはメッセージを問いかけてくる）として，移動方向右（R），範囲A1，G7と指定すればいい。ⓡの場合には移動方向上（U），範囲K35，R44となる。

図4

イ

ロ

DELETE

列または，行を削除するコマンドである。10行目を削除すると現在の11行目が10行に自動的に送られる。行iは，行i－1（i≧11）となる。列の場合も同様に，削除された列より右の列には，現在より一つ前の列名がつく。

行，列の変化に伴って式が入力されている場合，その中の座標も自動的に変換される。DELETEコマンドは不注意な使用によりデータを消失させる恐れがあるので，システムは実行前にYes, Noを問うメッセージを表示する。

INSERT

DELETEコマンドとは逆に，列や行を挿入するコマンドである。挿入された列や行の右，または下にある列，行の名称は変化する。

(3) フォーマットコマンド

WIDTH

ブロックの幅はDEFAULTで13文字となっているが，最大59字まで，このコマンドを使うことによって拡大できる。また，縦罫線などのためブロック幅を1文字まで縮小することも可能である。式の長さは最大60文字だが，ブロックには式が直接入力されるのではなく，計算結果が入力される。よってブロック幅を決めるとき式の長さは無視してよい。このコマンドによって決まったブロック幅はブロックカーソルのある列全体に適用される。WIDTHコマンドの活用により，見やすい表が作成可能だ。

FORMAT

このコマンドは各ブロックについて数値のフォーマット，すなわち桁数，小数点位置，3桁くぎりのカンマの有無，￥マークの有無を決定する。

たとえば初期状態のままで10／3などの計算を行うと，商は3.33333333333という形で表示される。FORMATコマンドで商の入るブロックを整数桁2，小数桁2のように指定すれば，商は3.33と表示される。この場合，コマンドのパラメータとして，＃＃.＃＃を入力する。

FORMATコマンドとWIDTHコマンドを有効に組み合わせれば，より表が見やすくなる。また￥マークありの指定をしておけば，金額表示のための別個の列を設けなくてすむ。

FMCALCについて，その特徴の一部を紹介しましたがいかがでしたでしょうか。誌面の都合で，画面処理コマンド，計算コマンド，入出力コマンドなどが紹介できませんでしたが，これらのコマンドについては次回で触れる予定です。さらにFMCALCの応用例もあわせて紹介する予定です。

おもしろゲーム大募集 発表!!

おもしろゲーム　講評

昨年暮，㈱日本ソフトバンクはハード別情報誌Oh!シリーズの各誌上において「おもしろゲーム」を大々的に募集しました。

特賞賞金100万円をはじめ総額260万円にのぼるゲーム募集としては大イベントだったためか，反響も大きく，全国規模の応募があり，その総数は最終的に121を数えました。応募者の方々には厚くお礼申し上げます。

審査は当社技術室が担当，厳正の上にも厳正な検討を行い，特賞以下9作品を以下のように決定しました。

	作品	機種	氏名
特賞	TANAKAのフライトシミュレータ	PC-9801	田中明夫
1等	前線基地	X1	Elder Man
2等	P3Cオライオン	PC-8801	水野康治
3等	WALL ALIEN	PC-8001	加藤利勝
入選	人生ゲーム	PC-6001	荻野浩一郎
〃	モグラタイジ	HC-20	松山浩治
〃	TANK BATALION	MZ-80B	鈴木政信
〃	戦　艦	PC-8001	池谷裕之
〃	ROULETTE	PC-9801	越智義明

審査の第1の基準はもちろんゲームとしておもしろいかどうかです。技術室ではこれを，アイデアの斬新さ，ストーリー性，操作性，スピード，画面構成，さらに効果音の成果等を個別に評価し，それをもとに総合的に優劣を決定しました。

それではここで入選した上位3作品のゲームを簡単に紹介しましょう。

特賞の「TANAKAのフライトシミュレータ」PC-9801のGDCをフルに活かしたリアルタイタイムのフライトシミュレータ。飛行感覚は既存のゲームとは比べものにならない程優れています。審査員全員一致で賞讃の意を表する次第です。くわしい説明とリストはOh!PC6月号に掲載されております。

1等の「前線基地」は，X1のPCG，PSGの機能を十分にひき出したALL BASICのゲームです。内容は基地を目指して攻めてくる戦車軍団を対戦車砲で撃破し，基地を守りぬくというものです。ストーリー性のある展開と美しい画面，それに効果音の利用を高く評価されました。詳しくはOh!MZ7月号でプログラムリストと共に紹介するつもりです。

2等の「P3Cオライオン」は初心者向けのシミュレーションゲームです。ソ連の潜水艦が津軽海峡を通過しないよう，P3Cで哨戒しながら駆逐艦で攻撃するという設定になっています。ルールが繁雑でないため誰でも楽しめるシミュレーションゲームとしてまとまっている点が評価されました。

応募作品の中には以上の9作品以外にも楽しいゲームがたくさんありました。今後Oh!シリーズでは紙面の許す限り，ご紹介していきたいと考えています。ご期待ください。

なお，残念ながら，今回はFM用作品で優れたものがなく，選外となりました。FMユーザーの奮起を望みます。

㈱日本ソフトバンク編集部技術室

for FM-7,8,11

CP/M-80とその使い方

林　剛正

最近のパーソナルコンピュータのカタログをみると，一昔と違って，BASIC言語よりも使用できるOS(オペレーティングシステム）に宣伝の重点をおいているようだ。

これは，パーソナルコンピュータはもはや，必要なものすべて自分一人で作ってしまうホビイストの時代に別れを告げ，だれでも既製のソフトウェアで自分のしたいことができる普及期に突入した現れである。そのため，人より使えるOSをたくさん持つことは，人より多くのソフトウェア供給源を持つことであり，人より情報の交換がやりやすいことでもある。この点において，FMシリーズのユーザーは，現時点において最高の幸せ者である。FM-7,8は，OS-9とFLEX，CP／M-80などが使える。FM-11となると，このうえにMS-DOS，CP／M-86あるいはコンカレントCP／M-86が使用可能である。しかし，これらのOSをすべて個人で導入することは財政上なかなか難しいし不経済でもある。また，ユーザーのコンピュータに関する知識の深浅によって、使いこなせるものと使いこなせないものがでてくる。一般的にいえば，OSの機能が高ければ高いほど，それだけ，ユーザーにより高度な専門知識が要求される。したがって，専門家の間で評判の良いOSが必ずしも，一般のユーザーにとって使いやすいものとは限らない。

もう一つ重要なことは，そのOS上で使える言語やユーティリティの豊富さである。いくらOSそのものが立派であろうと，しっかりした言語やユーティリティ，アプリケーションプログラムがなければ，まるでガソリンのない車みたいに，使えない。等々のことを考えれば，CP／M-80は，現時点において最高のものである。CP／M-80は，80系(8080,Z80)のためのOSであるが，幸いにもFMシリーズはZ80カードの追加により，6809マシンからZ80マシンに変身できる。したがって，FMのユーザーでも，CP／M-80の上に走る膨大なソフトを利用できる。

そこで，FMのCP／M-80(とはいっても，一般的なCP／M-80と同じ）を紹介し，その上で走るいくつかの高級言語やユーティリティなどについてもふれてみよう。

CP/M-80とは

CP／MはControl Program for Microprocessorsの略称であり，ユーザーの入力を分析し実行するとかフロッピーディスクを管理するなど，80系マイクロコンピュータの開発用モニタコントロールプログラムである。

1 CP／Mのメモリ構成

CP／Mは論理的に以下の4つの部分に分割されている(図1)。

1) BIOS (Basic I／O System)

入出力装置と情報交換を行うための最も基本的なルーチンの集まりであり，この部分だけ，ハードウェアに依存している(すなわち，この部分のプログラムをCP／Mの作成者が自分で作らなければならない)。

2) BDOS (Basic Disk Operating System)

ディスク装置と情報交換を行うための基本的なルーチンの集まりである。

3) CCP (Console Command Processor)

コンソールから入力されるコマンドを処理するルーチンの集まりである。

4) TPA (Transient Program Area)

実行プログラムが格納されるメモリ領域である。普通は，$0100番地からはじまるメモリ領域である。

＊　＊　＊

以上のように，メモリが分割使用されているが，CBASEやFBASEの番地を調整することにより，TPA領域の大きさも変わる。ふつう，××K CP／Mというのは，××KバイトのTPA（すなわち，ユーザーが自由に使用できるフリーメモリエリア）があることを意味する。したがって，TPAサイズより大きなプログラムサイズを持つアプリケーションプログラムは，実行できない。FDOS部は，システム開発をしないユーザーにとって，その内容がわからなくてもよいのでここでは省略する。

図1　CP／Mのメモリ構成

FDOS
フロッピーディスク
オペレーティングシステム

FBASE：

CCP
コンソールコマンドプロセッサ

CBASE：

TPA
トランジェントプログラム領域

TBASE :$0100
BOOT :$0000

BOOT
ブートストラップローダ

BIOS

CONST
コンソールステータス入力

CONIN
コンソールへ1文字出力

CONOUT
コンソールから1文字入力

その他ユーザーが定義したもの

BDOS
（各ディスクに64個までのファイルが格納できる）

SEARCH
名前により、ディスク内のファイルをみつける

OPEN
それ以後の操作のためファイルをオープンする

CLOSE
操作後にファイルをクローズする

READ
ファイルからレコードを読み出す

WRITE
ディスクにレコードを書き込む

RENAME
ファイルの名前を変更する

SELECT
使用するディスクドライブを選択する

2 CP／Mのコマンド構造

CP／Mでは，そのファイルマネージメント部により，ソース型式や実行型式などの多数の異なるプログラムファイルを収納できる。CP／Mのコマンドは，CCPの内部コマンド（ビルトインコマンド）と，ディスクからTPAにロードし実行されるプログラム（トランジェントコマンド）からなる。

ビルトインコマンドは，常時，メモリ上のCCP内に存在する。TPA領域をできるだけ大きくするために，ビルトインコマンドは，ディスクとのやり取りをする最小限のものしかない。それらを紹介する前に，まずCP／Mのファイル名について説明する。

ファイル名は，プライマリネームとセカンダリネームから成る（図2）。プライマリネームは，普通のファイルネームである。セカンダリネームは，CP／M上で走る多くのプログラムの属性を示すためのものである。もちろん，ユーザーが指定することができる。しかし，一般的には，アプリケーションプログラムによって，このセカンダリネームが自動にセットされる（図3(a)）

図2　CP／Mのファイルネーム構造

内部コマンドは，以下のようなものがある。

(1)　DIR (DIRectory)

ディスク内のすべてのファイル名をリストアップする（図3(b)，図4）。

(2)　ERA　(ERAse)

指定したファイルを削除する（図4）。

(3)　REN (REName)

ファイル名を変更する（図4）。

(4)　TYPE (TYPE)

ファイル（高級言語やアセンブラのソースファイルなどASCIIコードのファイル）をコンソールに出力する（図4）。

(5)　SAVE (SAVE)

ディスクにTPAの内容（プログラムなど

図3 (a)ディレクトリの出力およびその解説

	ファイル型名	ファイルの種類(その中身として)
1	ASM	アセンブラ言語プログラムファイル
2	BAS	BASIC言語プログラムファイル
3	BAK	バックアップファイル
4	COM	機械語プログラムファイル
5	DAT	データファイル
6	HEX	インテルHEX形式ファイル
7	INT	BASIC中間ファイル
8	PRN	プリントファイル

CP/Mで利用できるディスケット1枚には最大64個のファイルを格納することができる。各ディスケットにはディレクトリと呼ばれる表が一つ記憶されている。ディレクトリに格納されている内容は，そのディスケット中の全てのファイル番号，ファイル主名，ファイル型名，ファイルの大きさ，実際のデータが格納されているトラック番号とセクタ番号である。

図3 (b)

```
A>dir b:
B: IFINSTL  COM : IFCONFIG COM : IFCONFIG DAT : MBASIC   COM
B: BASLIB   REL : BASCOM   SUB : BASCOM   COM : F80      COM
B: FORLIB   LIB : FORLIB   REL : CREF80   COM : LIB      COM
B: BRUN     COM : BCLOAD       : L80      COM : LIB80    COM
B: DISK     DOC : MAC      COM : SAMPLE   ASM : I8085    LIB
B: Z80      LIB : Z80      DOC : INTER    LIB : TREADLES LIB
B: BUTTONS  LIB : SIMPIO   LIB : SEQIO    LIB : STACK    LIB
B: DSTACK   LIB : COMPARE  LIB : NCOMPARE LIB : WHEN     LIB
B: DOWHILE  LIB : SELECT   LIB : ZSID     COM : HIST     UTL
B: TRACE    UTL : OBSLIB   REL : M80      COM : RANTEST  BAS
B: RANTEST  REL : RANTEST  COM : RANTEST  ASC : PIP      COM
B: STAT     COM : SUBMIT   COM : XSUB     COM : ASM      COM
B: DDT      COM : LOAD     COM : DUMP     COM : D        COM
B: COMPILE  SUB : WM       COM : LSI      ASM : HAZ      ASM
B: SOL      ASM : BEE      ASM : P-E      ASM : WM       HLP
B: HAZ      HEX : LSI      HEX : SOL      HEX : BEE      HEX
B: P-E      HEX
A>
```

ビルトインコマンドであるDIRを使って，ディスクB上にあるファイルをリストアップ。セカンダリネームでそのファイルの属性がわかる(図3(a)参照)

のオブジェクトコード）をセーブする。また，CP/Mは原則上，英大文字と小文字を区別しないので，コマンドおよびファイル名は好きなモードで入力できる。しかし，CP/Mからのアウトプットは，すべて英大文字モードである。

*　*　*

ところが，このビルトインコマンド以外に，ユーザーがコマンドを追加できる。それらは，トランジェントコマンドである。実をいうと，このトランジェントコマンドというのは，実行可能のオブジェクトファイルである。つまり，CCPは，まずユーザーが入力した文字列を分析し，それがビルトインコマンドに相当するなら，そのコマンドを実行するが，もしビルトインコマンドでなかったら，ディスク上にそれのファイル（実行可能のオブジェクトコードからなるプログラムファイル）をTPAにロードし実行（普通は，$0100番地から)する。こういうわけで，ユーザーが必要と思われるものを作れば，コマンドが無限に拡張できる。市販の高級言語コンパイラやユーティリティもすべてトランジェントコマンドである。しかし，CP/Mのシステムのなかに，CCPの下で実行するように定義されたトランジェントコマンドはいくつかある。

1) STAT

現在ログインされているディスク上のファイルの大きさなどの情報を示す。あるいは，周辺デバイスの割り付けをする。

2) DUMP

ファイルの内容を16進数でコンソールにダンプする。

3) MOVCPM

ある特定のメモリサイズのCP/Mシステムを作成する。

4) SYSGEN

新しいCP/Mのシステムディスケットを作る。

5) SUBMIT, XSUB

コマンドのファイルをバッチ処理用にする。

6) ED

CP/Mのテキストエディタである。ソースファイルなどのASCIIコードファイルを作成する。

7) ASM

Digital Researchの8080アセンブラを実行し，8080のアセンブラ・ソースファイルを，機械語のファイルに翻訳する。

8) LOAD

Intel hex machine code formatでファイルを読み，TPA内にロードされて実行可能な型(.COM)のファイルを作る。ASMとLOAD，SAVEによって，新しいトランジェントコマンドファイルが作成できる。

9) DDT

ダイナミックデバッキングツールであり，会話形式によりプログラムのテスト，およびデバッグを行う。

10) PIP

ファイルの転送を行う。

*　*　*

このように，機能の高いトランジェントコマンドの標準装備によって，CP/MはOSであると同時に，システム開発環境をも提

図4　DIR,ERA,TYPE,REN ビルトインコマンドの実行

```
B>DIR  ←いまディスクB上にあるファイルをリストアップ
B: DEMO1     BAS : 3D-GRA    BAS : SETCHA    BAS : S-TREK    BAS
B: GOLF      BAS : SAMPLE    BAS : COMMAND   BAS : ADCB      BAS
B: SCALE     BAS : LRU       BAS : GRAPHIC   BAS : TEST      BAS
B: TESTX     BAS
B>REN SUPER-TK.BAS=S-TREK.BAS ←S-TREK.BASというBASICのソースファイルの名前をSUPER-TK.BASに変更
B>DIR ←DIRで確かめてみる
B: DEMO1     BAS : 3D-GRA    BAS : SETCHA    BAS : SUPER-TK BAS ←ちゃんと変わった
B: GOLF      BAS : SAMPLE    BAS : COMMAND   BAS : ADCB      BAS
B: SCALE     BAS : LRU       BAS : GRAPHIC   BAS : TEST      BAS
B: TESTX     BAS
B>ERA GOLF.BAS ← GOLF.BASというファイルを消去する
B>DIR ←DIRで確かめてみる、確かにGOLF.BASがなくなった
B: DEMO1     BAS : 3D-GRA    BAS : SETCHA    BAS : SUPER-TK BAS
B: SAMPLE    BAS : COMMAND   BAS : ADCB      BAS : SCALE     BAS
B: LRU       BAS : GRAPHIC   BAS : TEST      BAS : TESTX     BAS
B>ERA TEST?.BAS このマークは任意のキャラクタを表す。すなわち、このパターンにマッチするすべてのファイルを消去する
B>DIR
B: DEMO1     BAS : 3D-GRA    BAS : SETCHA    BAS : SUPER-TK BAS
B: SAMPLE    BAS : COMMAND   BAS : ADCB      BAS : SCALE     BAS
B: LRU       BAS : GRAPHIC   BAS
B>A: ←ログインディスクをディスクAに変更
A>DIR *.ASM このマークは任意長のストリングを表わす。すなわち、ASM属性のファイルのみをリストアップ
A: DUMP      ASM : BIOS      ASM : CBIOS     ASM : DEBLOCK   ASM
A: LSI       ASM : HAZ       ASM : SOL       ASM : BEE       ASM
A: P-E       ASM : IF800     ASM
A>TYPE DUMP.ASM ←DUMP.ASMというアセンブラのソースファイルをコンソールに出力する
;       FILE DUMP PROGRAM, READS AN INPUT FILE AND PRINTS IN HEX
;
;       COPYRIGHT (C) 1975, 1976, 1977, 1978
;       DIGITAL RESEARCH
;       BOX 579, PACIFIC GROVE
;       CALIFORNIA, 93950
;
        ORG     100H
BDOS    EQU     0005H   ;DOS ENTRY POINT
CONS    EQU     1       ;READ CONSOLE
TYPEF   EQU     2       ;TYP
                ≀≀

;
GO:     ;READ THE BYTE AT BUFF+REG A
        MOV     E,A     ;LS BYTE OF BUFFER INDEX
        MVI     D,0     ;DOUBLE PRECISION INDEX TO DE
        INR     A       ;INDEX=INDEX+1
        STA     IBP     ;BACK TO MEMORY
;       POINTER IS INCREMENTED
;       SAVE THE CURRENT FILE ADDRESS
        LXI     H,BUFF
                ≀≀
```

供してくれる。

CP/M上のアプリケーションプログラムの使い方

一般的なユーザー，すなわち，自分がプログラムを開発しないで，ただ市販のアプリケーションプログラム（たとえば，データベースやスーパカルクなど）を使いたい場合には，システムプロンプト（">"）のあとにそのファイル名を入力すれば，たとえば，スーパカルクなら

A>SC

とタイプすれば直ちに使用できる。

マイクロソフト社のBASICインタプリタを使いたいなら，

A>MBASIC↵

と下線部をタイプすれば，F-BASICとほぼ同じようなBASICの世界に入れる。

このように普通のプログラムについては，そのプログラムの起動方法さえわかれば，CP／Mのオペレーションが理解できなくて

も大丈夫だ。しかし，インタプリタやユーティリティと違って，アセンブラと高級言語コンパイラの使い方にやや手間がかかるものである。その手順は**図5**の示すとおりである。

ED.COMなどのトランジェントコマンドファイルは，標準CP/Mのシステムディスケットのなかにすでにあるものばかりである。それらを使えば，新たなトランジェントコマンドを作ることは簡単に可能である。しかし，そのアセンブラは8080のものであるため，CPUのZ-80の性能を充分に引き出せない。したがって，Z-80アセンブラを使用した方がよい。そこで，Z-80アセンブラの中でも有名であるマイクロソフト社のMACRO-80を使って，適当なトランジェントコマンドファイルを作ってみる。

プログラムは，入力した一文字を255回画面にエコーバックするものである。これ自体は，コマンドとしては使えないものであるから，プログラムよりもそのできあがるまでの過程に注意されたい。ソースファイルのEX 1. MACは，エディタプログラム(ED. COMやWM. COM）を使って作成したもので，属性の.MACはそのファイルがM80.COMの関連ファイルであることを示している。**図6**からわかるように.COM属性のファイルは，そのプライマリネームを入力するだけで，メインメモリにロードされ即実行されるのである。また，すべてのトランジェントコマンドは，CP/Mというモニタの上で走るため，^C(コントロールキーとCキーを一緒に押す）を押すことにより，いつでもCP/Mのシステムモードに戻れる。

図5　トランジェントコマンドの作り方(a)　トランジェントコマンドの作り方(b)

図7は，C言語を使った，プログラム開発の過程を示している。マイコン上で走るC言語は，ほとんどUNIX上のC言語のサブセットである。本稿に使われた"BD Software C Compiler"のほかにも，"Tiny C"とか，"WHITESMITHS' C Compiler"などがある。後者のホワイトスミスCは，60K以上のCP/Mでないと動かないもので，FMのユーザーは，残念ながら現時点では使用できない。図にあるtest.Cというプログラムは，入力された文字列のアドレスおよび文字数を調べて逆順に出力するものである。このソースファイルの作成は，先と同じように，EDやWM(スクリーンエディタ)を使った。CC 1は，test.CというC言語のソースをコンパイルし，リロケータブルファイルTEST.CRLを作る。Clinkは，リンカであり，TEST.CRLに必要なルーチン（ライブラリより取り出す）をつなぎ，TEST.COMというオブジェクトファイルを作る。できあがったトランジェントファイルを実際に走らせてみると，**図7**のようになった。入力に，"This ␣ is ␣ a ␣ test ␣ program."の23文字を入れるつもりだったが，途中に一個所訂正（バックスペース＋新しい1文字）があったため，入力文字数が25となった。

以上のように，CP/Mの既有のトランジェントコマンドや高級言語，ユーティリティを使用することによって，ユーザーのアプリケーション開発が簡単にできる。もちろん，BASICインタプリタに比べて，アセンブラやコンパイラを使ったプログラム開発は複雑であるが，できあがったトランジェントファイルは，機械語プログラムであるため，BASICインタプリタに比べて実行速度がはるかに速い。

CP/Mのシステムコール

CP/Mを使ってプログラムを開発する際に，高級言語によるなら問題ないが，アセンブラでやると，キーボードの入力ポート番号とか，画面への出力云々などを考えなければならない。しかし，CP/MはI/O関係の基本的ルーチンをシステムでサービ

図6　M80を使ったプログラムの作成


```
A>PIP B:=A:EX1.MAC ←トランジェントのコマンドpipを使って、ディスクBにディスクA上のEX1.MACというファイルを転送する
                     (すなわち、コピー)
A>B:                 }
B>DIR                } ログインディスクを切り換えて、ディスクB上にEX1.MACがあることをDIRで確かめる
B: EX1      MAC      }
B>^C
B>A:M80 ←ディスクAからマイクロソフト社のマクロアセンブルM80.COMをロードする

*EX1,EX1=EX1/Z ←EX1.MACをアセンブルし、リロケータブルファイルEX1.RELおよびリストファイルEX.PRNを作成する

No Fatal error(s)

*^C ←M80.COMの実行を終了させCP/Mシステムに戻る
B>A:L80 ←ディスクAからマイクロソフト社のリンクローダLINK-80をロードする

Link-80  3.44  09-Dec-81  Copyright (c) 1981 Microsoft

*EX1,EX1/N/E ←EX1.RELよりEX1.COMというオブジェクトファイルを作る。そのあとCP/Mシステムに戻る

Data    0100    0141    <    65>  }
                                  } オブジェクトファイルについての情報
47069 Bytes Free                  }
[0000   0141          1]          }

B>DIR ←確かめてみる
B: EX1      MAC : EX1       PRN : EX1       REL : EX1       COM
                                                  できた！
B>EX1 ←さっそく実行
*A:AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAA
*0:00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000
*
B>TYPE EX1.MAC ←EX1.MACのソースファイルの中身をリストアップ
        .Z80
        ASEG
        ORG     100H
;
START:: LD      C,2
        LD      E,"*"
        CALL    0005H   ←BIOS Call
;
        LD      C,1
        CALL    0005H
;
        LD      (SAVE),A
        CP      12H             ;IF ^R THEN REBOOT SYSTEM
        JP      NZ,ECHO
        LD      C,0
        CALL    0005H
;
ECHO::  LD      C,2
        LD      E,":"
        CALL    0005H
        LD      B,0FFH
OUT::   LD      C,2
        LD      A,(SAVE)
        LD      E,A
        PUSH    BC
```

```
        CALL     0005H
        POP      BC
        DJNZ     OUT
;
        LD       C,2
        LD       E,13
        CALL     0005H
        LD       C,2
        LD       E,10
        CALL     0005H
;
        JP       START              ;NEXT
;
SAVE::  DEFB     1
;
        END

B>ERA *.* ←—いまログイン中のディスク(=ディスクB)上のすべてのファイルを削除する
ALL (Y/N)?Y ←—あまりにも危険なので、CP/Mの方から確認を要求する
B>dir b:
NO FILE ←—何にもなくなった
B>
```

表1　システムの機能の要約

注＊：終了時の状態では、次のようになっている。レジスタA＝レジスタL　レジスタB＝レジスタH

機能コード	機能	入力パラメータ	終了時の状態
0	システムリセット	なし	なし
1	コンソールからの入力	なし	A＝ASCII文字
2	コンソールへの出力	E＝ASCII文字	なし
3	リーダからの入力	なし	A＝ASCII文字
4	パンチへの出力	E＝ASCII文字	なし
5	リストへの出力	E＝ASCII文字	なし
6	コンソールによる直接入出力	定義参照	定義参照
7	入出力データバイトの取り出し	なし	A＝IOBYTE
8	入出力データバイトのセット	E＝IOBYTE	なし
9	文字列のプリント	DE＝BUFFER	なし
10	コンソールバッファへの読み込み	DE＝BUFFER	定義参照
11	コンソールの状態信号の取り出し	なし	A＝00／FF
12	バージョン番号の取り出し	なし	HL＝バージョン番号＊
13	ディスクシステムのリセット	なし	定義参照
14	ディスクの選択	E＝選択するディスク番号	定義参照
15	ファイルのオープン	DE＝.FCB	A＝ディレクトリコード
16	ファイルのクローズ	DE＝.FCB	A＝ディレクトリコード
17	最初のデータのサーチ	DE＝.FCB	A＝ディレクトリコード
18	次のデータのサーチ	なし	A＝ディレクトリコード
19	ファイルの削除	DE＝.FCB	A＝ディレクトリコード
20	シーケンシャルな読み出し	DE＝.FCB	A＝エラーコード
21	シーケンシャルな書き込み	DE＝.FCB	A＝エラーコード
22	ファイルの作成	DE＝.FCB	A＝ディレクトリコード
23	ファイルの名前のつけ直し	DE＝.FCB	A＝ディレクトリコード
24	ログインベクトルの取り出し	なし	HL＝ログインベクトル＊
25	選択されているディスク番号の取り出し	なし	A＝選択されているディスク番号
26	DMAアドレスのセット	DE＝.DMA	なし
27	アロケーションアドレスの取り出し	なし	HL＝アロケーションアドレス
28	書き込み保護ディスク	なし	定義参照
29	R/O(読み出しのみ可能)ベクトルの取り出し	なし	HL＝R／Oベクトル
30	ファイル属性のセット	DE＝.FCB	定義参照
31	ディスクパラメータアドレスの取り出し	なし	HL＝ディスクパラメータブロックのアドレス＊
32	ユーザーコードのセットまたは取り出し	定義参照	定義参照
33	ランダムな読み出し	DE＝.FCB	A＝エラーコード
34	ランダムな書き込み	DE＝.FCB	A＝エラーコード
35	ファイルの大きさの計算	DE＝.FCB	r0　r1　r2
36	ランダムレコード番号のセット	DE＝.FCB	r0　r1　r2

（デジタルリサーチ社CP／Mマニュアルより）

スする（**表1**）ので，機種がかわったからといって，プログラムを書き替えることは無用である。だから，たとえばコンソールに1文字を出力したいとき

1）Cレジスタに，〝コンソールへの出力〟の機能コード2を代入

2）Eレジスタに，出力したい文字のASCIIコードを代入

3）アドレス$0005をコールする。

とアセンブラでプログラムすれば，入出力ポートやメモリのことを考えなくてすむ（**図6**のアセンブラのソースリストを参照)。このように必要なパラメータを該当するレジスタにセットし,Cレジスタに機能コードのセットおよび$0005のアドレスをコールすることで，I／O関係のルーチンが簡単にできることは，CP／Mの強みである。

CP／Mが管理する入出力機能やディスクのコントロールを，ユーザーに開放することが，システムサービスであり，それらをプログラム中で使用することをシステムコールという。FMのBIOSコールとやや似ている。違っているのは，FMのBIOSはFMしか使用できなく，CP／MのBIOSは，ハードが違ってもプログラムがちゃんと動作する。

終わりに

CP／Mについての専門書は，最近になって急に多くなって，ほとんどの本屋さんには必ず置いてあるので，興味のある方は，ぜひそれらも読むとよいだろう。本稿は，CP／Mについての紹介なので，初心者にとって難しいと思われる概念をなるべく省いたため，ややかけ足となってわかりづらいと

図7　BDS-Cコンパイラを使ったプログラムの作成

```
A>type test.c    ←─ビルトインコマンドTYPEを使って、test.Cのソースファイルの中身をコンソールにダンプ
#define EOL        '\n'
#define BUFFER   80

main ()
{
        int i;
        char c[BUFFER];
        char *p;

        i = 0;
        p = &c[0];
        printf("Address start : %4x\ninput>>", p);

        while ((c[++i] = getchar()) != EOL);

        p = &c[--i];
        printf("Address end : %4x\n", p);
        printf("%3d characters\n", i);

        while ((i--) != 0)
                printf("%c", *p--);
        printf("\n");
}
                                        } ソースファイル＝Cのプログラム
A>cc1 test.c
BD Software C Compiler v1.46  (part I)
  39K elbowroom
BD Software C Compiler v1.46 (part II)
  34K to spare
                                        } BDS Cコンパイルを使って
                                          コンパイルする
A>clink test ←─リンカClink.COMを使って、オブジェクトファイルtest.COMを作る
BD Software C Linker   v1.46
Linkage complete
  45K left over
A>dir test.*
A: TEST     LIN : TEST     CRL : TEST     COM : TEST     C
A: TEST     BAK
                                  オブジェクトファイル   ソース
                                        ‖
                                  トランジェントコマンド
A>test ←─実行
Address start : E3AC
input>>This is a test program.
Address end : E3C5
 25 characters
.margorp tret a si sihT

A>
```

思うが，CP／Mのイメージだけでもわかってもらえたら幸いである。また，CP／Mを購入すると，厚く親切な（？）マニュアルも付いてくるので，心ある方は，勉強してほしい。

CP／MというOSは，もちろん時代遅れの点もあるが，全体的にはよくまとまっており，シンプルなのでとても使いやすいものである。そして，今日まで蓄積されてきた膨大なソフトウェアライブラリによって，8ビットマイコンの標準OSの王座に居続けるであろう。マイコンのBASICインタプリタにあきた方は，ぜひCP／Mの世界に来てほしい。そして，なぜ自分がもっと早くからCP／Mを導入しなかったかを後悔するであろう。

くやしさいっぱいのFM-8ユーザーのために

FM-8の命令を増やそう！

FM-7が発売されて，FM-8のユーザーのみなさんは，その価格，性能にさぞくやしい思いをしたことでしょう。そこで，今回は少しでもFM-7に近づけようということで，F-BASIC(V.1.0)の拡張を試みてみました。

Minky Software
企画部
田近 智彦

概　要

F-BASIC では後々ステートメントの拡張ができるように，あらかじめ特別なワークエリア(GAP)が設けられています(**表1**参照)。これらの内容を適当に変更することにより，ステートメント，関数などの拡張ができます。ちなみに FM-7 が搭載している F-BASIC(V. 3.0)はこの方法でCHAIN, PLAY などの命令を拡張しています。

フローチャート　1

表１

アドレス	意　味	参　考
$01F9	ステートメントの拡張数	①ROM-BASICを拡張するために用いる。
$01FA,B	キーワードテーブルの先頭アドレス	(DISKシステムで使用)
$01FC,D	処理ルーチンの先頭アドレス	②中間言語は$E8から割り当てられる。
$01FE	関数の拡張数	①について同上
$01FF,200	キーワードテーブルの先頭アドレス	関数の拡張に用いる。
$0201,2	処理ルーチンの先頭アドレス	
$0203	ステートメントの拡張数	③DISK-BASICを拡張するために用いる。
$0204,5	キーワードテーブルの先頭アドレス	
$0206,7	処理ルーチンの先頭アドレス	④中間言語は，$EEから割り当てられる。
$0208	関数の拡張数	③について同上。
$0209,A	キーワードテーブルの先頭アドレス	
$020B,C	処理ルーチンの先頭アドレス	

拡張の手順

ここでは，ステートメントを拡張する場合についてのみ説明しますが，関数の拡張も基本的に同じです。

ステートメントの拡張は次の手順で行います。

①どのような働きをする命令を拡張するのか充分考える。

この時点で，書式（引数の数，形式，値の範囲など）を詳しく決定します。

②予約語拡張の定義をするプログラム（ドライバプログラム）を制作する。

先に概要のところで説明したワークエリアを書き換えるためのプログラムを制作します。

③新ステートメントの処理用プログラム（インタプリンタ）を制作する。

具体例

具体例として次の３つのステートメントを拡張してみました。それぞれ前項の手順に従って説明していきます。

①拡張する命令

○BELL

書式：BELL〔スイッチ〕スイッチ＝0or1

機能：F-BASIC の BEEP に同じ

なにも F-BASIC にある命令を拡張する必要はないのですが，ステートメントの処理の構造の例として挙げました。

○LPRINT

書式：LPRINT 式または文字式

機能：プリンタに対しても印字を行う。

つまり，プリンタおよび画面に対し双方に印字します。F-BASIC(V.2.0,

リスト1　BASICの拡張


```
01000                           TTL     BASIC ノ カクチョウ
01010                   *
01020                   * COPYRIGHT (C) BY Minky software.
01030                   *   PROGRAMMED BY Tomohiko Tajika.
01040                   *
01050              6000 START   EQU     $6000
01060 6000                      ORG     START
01070                   *
01080                   * SYSTEM Subroutine
01090                   *
01100              00D2 GET     EQU     $00D2
01110              00D8 GETCHR  EQU     $00D8
01120              01F9 EXNUM   EQU     $01F9     (DISK $0203)
01130              01FA KEYADR  EQU     $01FA     (DISK $0204)
01140              01FC PRCADR  EQU     $01FC     (DISK $0206)
01150              9740 SNERR   EQU     $9740
01160              9B03 ILLERR  EQU     $9B03
01170              92AB ERROR   EQU     $92AB
01180              D352 ACCPRT  EQU     $D352
01190              9D86 STRVAL  EQU     $9D86
01200              9E8F VALX    EQU     $9E8F
01210              9E49 VALB    EQU     $9E49
01220              9732 COTEST  EQU     $9732
01230              9F0D PRINT   EQU     $9F0D
01240              05AC LPFLG   EQU     $05AC
01250              9FDF MSGPRT  EQU     $9FDF
01260                   * カ ク チョ ウ ル ー チ ン
01270 6000 86      03   ENTRY   LDA     #3        ス テ ー ト メ ン ト ス ウ
01280 6002 B7      01F9         STA     EXNUM
01290 6005 30      8D 0012      LEAX    CMDTBL,PCR キ ー ワ ー ド テ ー ブ ル
01300 6009 BF      01FA         STX     KEYADR
01310 600C 30      8D 0022      LEAX    PROCES,PCR ショ リ ア ド レ ス
01320 6010 BF      01FC         STX     PRCADR
01330 6013 30      8D 0011      LEAX    MSG-1,PCR
01340 6017 BD      9FDF         JSR     MSGPRT    メッ セ ー ジ プ リ ン ト
01350 601A 39                   RTS
01360                   *
01370 601B         42   CMDTBL  FCC     'BELフ'   BELL
      601C         45
      601D         4C
      601E         CC
01380 601F         4C           FCC     'LPRINヤ' LPRINT
      6020         50
      6021         52
      6022         49
      6023         4E
      6024         D4
01390 6025         54           FCC     'TESヤ'   TEST
      6026         45
      6027         53
      6028         D4
01400                   *
01410 6029         B6   MSG     FCC     'カクチョウ OK'
      602A         B8
      602B         C1
      602C         AE
      602D         B3
      602E         20
      602F         4F
      6030         4B
01420 6031         00           FCB     0
01430                   *
01440 6032 81      EE   PROCES  CMPA    #$EE      DISK ?
01450 6034 25      02   6038    BLO     PRC1      NO.
01460 6036 80      06           SUBA    #6
01470 6038 80      E8   PRC1    SUBA    #$E8
01480 603A 27      09   6045    BEQ     BELL      YES.
01490 603C 4A                   DECA              LPRINT ?
01500 603D 27      26   6065    BEQ     LPRINT    YES.
```

```
01510  603F 4A                        DECA             TEST ?
01520  6040 27    31    6073          BEQ     TEST     YES.
01530  6042 7E    9740         PROT   JMP     SNERR    ｼﾝﾀｯｸｽｴﾗｰ
01540                          *
01550  6045 9D    D2           BELL   JSR     GET      SKIP statement code.
01560  6047 26    05    604E          BNE     BELL1
01570  6049 86    07                  LDA     #7       BELL CODE
01580  604B 7E    D352                JMP     ACCPRT
01590  604E BD    9E49         BELL1  JSR     VALB
01600  6051 86    01                  LDA     #1       A=1
01610  6053 5D                        TSTB             B=0 ?
01620  6054 26    02    6058          BNE     BELL2    NO.
01630  6056 20    09    6061          BRA     BELL4    YES.
01640  6058 C1    01           BELL2  CMPB    #1       B=1 ?
01650  605A 27    03    605F          BEQ     BELL3    YES.
01660  605C 7E    9B03                JMP     ILLERR   ｴﾗｰ
01670  605F 86    81           BELL3  LDA     #$81     A=$81
01680  6061 B7    FD03         BELL4  STA     $FD03
01690  6064 39                        RTS              ｼｮﾘｼｭｳﾘｮｳ
01700                          *
01710  6065 86    01           LPRINT LDA     #1
01720  6067 B7    05AC                STA     LPFLG
01730  606A 9D    D2                  JSR     GET
01740  606C BD    9F0D                JSR     PRINT
01741  606F 7F    05AC                CLR     LPFLG
01742  6072 39                        RTS
01750                          *
01760  6073 9D    D2           TEST   JSR     GET
01770  6075 BD    9D86                JSR     STRVAL
01780  6078 34    14                  PSHS    B,X
01790  607A BD    9732                JSR     COTEST   ｺﾝﾏﾁｪｯｸ
01800  607D BD    9E8F                JSR     VALX
01810  6080 9D    D8                  JSR     GETCHR   STATE MENT END ?
01820  6082 26    BE    6042          BNE     PROT     NO. SYNTAX ERROR
01830  6084 8C    04D2                CMPX    #1234    ｱﾝｼｮｳﾊﾞﾝｺﾞｳ
01840  6087 27    05    608E          BEQ     TEST1
01850  6089 C6    3E           ERR    LDB     #62      error code=62
01860  608B 7E    92AB                JMP     ERROR
01870  608E 35    14           TEST1  PULS    B,X
01880  6090 C1    0A                  CMPB    #KEYEND-KEY
01890  6092 26    F5    6089          BNE     ERR
01900  6094 33    8D 000A             LEAU    KEY,PCR KEY WORD
01910  6098 A6    C0           LOOP   LDA     ,U+
01920  609A A1    80                  CMPA    ,X+
01930  609C 26    EB    6089          BNE     ERR
01940  609E 5A                        DECB
01950  609F 26    F7    6098          BNE     LOOP
01960  60A1 39                        RTS
01970                          *
01980  60A2       49           KEY    FCC     'I LOVE YOU'
       60A3       20
       60A4       4C
       60A5       4F
       60A6       56
       60A7       45
       60A8       20
       60A9       59
       60AA       4F
       60AB       55
01990             60AC         KEYEND EQU     *
02000             6000                END     START
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=6000
PROGRAM END   ADDR=60AB
PROGRAM ENTRY ADDR=6000
```

V.3.0)のLPRINTと違いますので注意してください。

○TEST

書式：TEST,キーワード文字式,暗証番号,

機能：キーワードの文字列と暗証番号を，あらかじめ定義してあるものと比較して，同じであれば次の命令へ進み，異なればError62(Protected Program）を発生し，プログラムの中断をする。

②ドライバプログラム

アセンブルリスト1（1270～1420行）を見

表2

文字	bit 7を たてた文字	文字	bit 7を たてた文字
A	チ	N	ホ
B	ツ	O	マ
C	テ	P	ミ
D	ト	Q	ム
E	ナ	R	メ
F	ニ	S	モ
G	ヌ	T	ヤ
H	ネ	U	ユ
I	ノ	V	ヨ
J	ハ	W	ラ
K	ヒ	X	リ
L	フ	Y	ル
M	ヘ	Z	レ

表3 ROM内サブルーチン

アドレス	意 味・使用方法など	アドレス (FM-7)
$00D2	テキストポインタをインクリメントした後，$00D8へジャンプ。	$00D2
$00D8	テキストから1文字GETしてAレジスタに入れる。そのとき余分なスペースは読み飛ばす。 ステートメントエンドの場合は，ゼロフラグがセットされる。	$00D8
$9740	シンタックスエラー処理ルーチン。	$92A0
$9B03	Illeagal Function Call 処理ルーチン。	$9663
$92AB	エラー処理ルーチン。Bレジスタにエラーコードを入れる。	$8DD1
$D352	ファイルに対して1文字出力を行う，とくに指定しなければCRTへ。 ファイルNo.は，$00BFにセット。	
$9D86	文字式評価。Bレジスタに文字数，Xレジスタに文字列の先頭アドレスがセットされる。	
$9E8F	式評価を行って，その値をXレジスタにセットしてリターンする。	$9A02
$9E49	式評価を行って，その値をBレジスタにセットしてリターンする。	$99BC
$9732	コンマがあるかどうかチェック，あればリターン。	$9292
$9FDF	1行出力。Xレジスタの内容+1のアドレスから$00がでるまでプリント。	
$9F0D	BASICのPRINT処理ルーチン。	
$05AC	($05AC) ≠ 0のとき，プリンタに対しても出力。	
$FD03	$81でBELL ON，$01でBELL OFF。	$FD03

プログラム 2

```
1000 OPEN "O",1,"SCRN:"
1010 I=PEEK(&H1F0)*256+PEEK(&H1F1)
1020 JT=PEEK(&H1F2)*256+PEEK(&H1F3)
1030 C=&H80
1040 PRINT#1,HEX$(C)"    ";
1050 A$=A$+CHR$(PEEK(I)AND127)
1060 IF (PEEK(I)AND128)=0 THEN I=I+1:GOTO 1050
1070 PRINT#1,A$;TAB(15);
1080 TT=PEEK(JT)*256+PEEK(JT+1)
1090 IF (TT>&H100 AND TT<&H200) OR TT>&H7000 THEN PRINT#1,HEX$(TT) ELSE PRINT#1
1100 JT=JT+2:C=C+1:I=I+1:A$=""
1110 IF C=&HE8 THEN END ELSE 1040
```

ていただければわかると思いますが，**表1**のアドレスを書き換えたあと，拡張されたことを示すメッセージを表示するようになっています。

キーワードテーブルは，他のステートメントとの区切りを示すために，最後の文字のbit 7をたてます(**表2**参照)。

③処理プログラムの制作

この部分は文章より実際のプログラムの方がわかりやすいと思いますので，**アセンブルリスト1**(1440～2000行)および**フローチャート**を参照してください。

ハード関係や，BASIC ROM内の式評価サブルーチンのアドレスを**表3**に示します。**表3**に示したほかにもたくさんのサブルーチンがありますが，各自，既に定義されているステートメントなどを逆アセンブルして調べてください。**プログラム2**を実行すると，各ステートメントの処理先アドレスがわかりますので利用してください。

実際の動作のさせ方

$6000～$60ABまで入力して，EXEC&H6000を実行するとカクチョウOKと表示され拡張命令を使うことができます。

かけ足で説明してしまったので，わかりにくい面もあったかと思いますが，アセンブルリストなどを参考にして，各自いろいろと試してみてください。

フローチャート 4

READER'S AREA

▶とにかく2号がでたのは喜ばしい。男はカイショウ! 3号，4号，……NEXT。(宝塚市・徳岡泰博・33)

▶『キャハァ〜〜ツ!』なにを間違えたのか創刊号の私のアンケートはがきが載ってしまい，顔から火が出るほど恥ずかしいのと同時に，部屋で1人でいつまでもニヤケていた。が次の日，悪しくも浪人生となる身が決定してしまい，もうひらき直ってGF募集します。(大田区・内山登志夫・18)

──浪人生となってしまった内山君に愛の手を! 我と思わん方はお便りください。編集部が愛の飛脚になります。(今度の入試はがんばろーね!!)

▶バックナンバーは存在するのでしょうか。1月にFM-7を購入，以来本屋へ行っても週刊FMはあれどOh/FMはない。私にとって幻の本となっております(だいいち2号からじゃマシン語入門とかわかりませんよ)。ところで早いことOh/FMが月刊になり，週刊にまでなってしまったら……マイコン無知の人がFM雑誌とまちがえることは十分考えられます。その日が楽しみですねえ。(文京区・樋口貴章・21)

──このほかにも創刊号に関するお問い合せを多数いただきました。創刊号は売り切れで在庫がないのです。しかしっ!! コピーサービスを始めました。詳細はこのコーナーのおしりに載ってます，よく読んでね。ところでOh!FMはFM雑誌だと思っている人がいるらしいですが，それは大きな間違いです。実はOh!FMはSM雑誌なのだ。本屋さんに行って，Oh!FMがコンピュータコーナーにあったらそ〜〜っとSMコーナーに戻しておいてください。

▶年がいもなくコンピュータ手習いを始めて0.5年。見る記事，読む記事サッパリ。ぜひ初歩の記事をおねがいします。イロイロ失敗しています。(鎌倉市・森章・60)

──森さんがんばってください。誰でも初めは初心者なんだ!

▶明日FM-7が届きます(3月25日)。今日FM-7が届きます(3月26日)。一昨日FM-7が届きました(3月28日)。(名古屋市・水野雅則・24)

▶Oh!FM創刊!! ちっとも知りませんでした。よく見ると季刊，もっとよく見ると隔月刊となっているではありませんか。なぜ月刊にしない!! ナニ，来年こそ? 責任者出てこい!! (茂原市・水柴宏之・31)

▶隔月となるそうで，非常に喜んでいます。早く月刊誌となるようにがんばってください。フロッピーディスクユニットの各社の比較があり参考になりました。次はぜひ，フロッピーディスケットの比較をお願いしたい。(広島市・山川憲二・32)

▶なかなかいい。早く月刊誌になるといい。ところで今度，FMシリーズプログラムコンテストをやってみてはどうでしょうか。もっとOh/FMを太らせましょう。(春日井市・伊藤泰久・19)

▶ぼくから見たOh/FMはやはり参考になる雑誌だった。これからは全周辺機器の活用記事などを載せてほしい。(登別市・渋谷秀悦・13)

▶子供はさっそくゲームプログラムを走らせている。(茅ヶ崎市・星川欣孝・44)

▶隔月発行の夢のかなった今，次の希望は冊子化である。そうすればFMシリーズも格があがる。月刊化の前にぜひとも。広告を増やして。(須坂市・清水和久・17)

▶いよいよ隔月刊誌となりましたね。おめでとうございます。この次は月刊誌となるのを期待しています。ぼくのような低レベルの人間にもわかるのでうれしいです。(豊田市・兵頭知樹・17)

▶お知らせ欄に「第3号より隔月刊誌になる」と書いてあった。さらに良い記事，情報を載せてほしい。(笠間市・山田正巳・28)

▶隔月刊化おめでとうございます。とても内容の濃い紙面に，ただひたすら感動しています。今後はFM-7のサウンド機構に関する記事を望みます。(函館市・清水晴夫・14)

──ありがとうございます。みなさんのご支援の声をうけて，Oh!FMはがんばります! よろしくおねがいします。

▶たいへん充実していてすばらしい。何よりFM専用というのがうれしい。ところで創刊号を何としても手に入れたいのですが何とかなりませんか。(名古屋市・広田健二郎・19)

──なりません。

▶もうすぐFM-7を買います。それで，先日マイコンフェアにひやかしに行ってきました。そこにはNEC，シャープ，富士通，他数社のマイコンがぎっしり。どれもすんばらしいデモをやっていました。そこで一組の親子を見かけました。息子はFM-7の前に立ち「これ，これ!」母親は何もわからない様子で「……全部でいくらかかるんで?」ようするに母親にはこの，キーボードのついた箱やテレビが何なのかわからず，係員の説明にもボーッとしていた様子。FM-7が出されて困るのは世の大蔵大臣ばかりのようですね。(しかし富士通は値引きませんねエ。かなりネバッタけれど?%を引き出せただけで，そこからは店員もダンマリ。しかたなくいろいろと密約をして購入を約束しました。そのうちの1つはいうまでもなくあの伊藤麻衣子ちゃんのポスター1組であります。どーだ!)(徳島市・渡辺晋二)

▶FM-7，11の発表後，FM-8の情報が少

なくなったのがさびしい！ 私は最近FM-8を買ったのですが，うまく使いこなせません。FM-8の情報をたくさん載せてほしい。（所沢市・後藤勇治・34）

▶はっきりいってあまり面白くありません。マシン語入門にたいへん興味があったのにむずかしすぎた。よく考えてください。FM-7をほめてFM-8を持ってる人がうれしいわけありません。（因島市・小丸芳正・15）

▶FM-7や11はけっこうですのでFM-8の特集してください。ゲームプログラム特集なんかぜひやってください。（橿原市・柳原 通・16）

▶FM-8の記事が少ない。7ばかりひいきしないで。（姫路市・谷口和生・16）

──別に7をひいきしているわけではないのです。こちらとしては7も8も11も均等にとりあげているつもりですが……。次に紹介する意見は少々辛口ですが，ちょっと考えさせられます。

▶前略，今回初めてOh! FMを買いました。第2号の投稿にFM-7の出現によって「FM-8の存在価値はなくなった」と言っておられますが，私はそう思いません。FM-7の悪口は言いたくないが，肥満化しているBASICになおかつじゃまなコマンドをつけ単にCPUクロックを倍速にして売っているだけである。FM-8のユーザーはFM-7のそれよりも2年近くの経験を持っているのです。ソフトウエアの技術は何よりも経験がものをいいます。FM-8のユーザーは，ハード・ソフトの両面において幾多の苦労を強いられてきました。その結果，多大なハード・ソフトの知識を得ることができました。FM-8の倍速化，BASICインタプリタのバージョンアップ（ユーザーレベル）……このようにFM-8のユーザーはその不満なマシンゆえに貴重な知識を自分のものにしたのではないでしょうか？ FM-7の出現に不満を言う人はFM-8を十分に活用できなかった・勉強をほとんどしなかった・BASICにしがみついていた人たちではないでしょうか？ FM-7の出現に不満を言うのはやめましょう。それは自分を「バカ」だと言っていることと同じです。そのような人はもっと勉強してください。（泉佐野市・芝野 晃）

▶1つ提案。Oh! FMのマスコットを作ったらどうでしょうか。そうすればOh! FMの固いイメージがなくなるのではないでしょうか。（泉市・笠松信隆・16）

──きゃ〜〜，FMにだってマスコットはあるんですよ。エリアのタイトルのイラストを見てください。もぐらがいるでしょう？ もぐらがマスコットなのだ。だれかもぐらに可愛い名前をつけて〜〜！

▶読者の中で不要，あるいは余分な創刊号がありましたら倍額で引きとります。送料はすべてこちらでもちますので，助けると思ってどうかおゆずりください。（〒174 板橋区常盤台4－4－1 竹林荘 菅家裕幸 TEL 03－937－2272）

パソコンFM祭り

7月8日(金)から10日(日)の3日間，富士通㈱主催の「パソコンFM祭り」が池袋サンシャインシティ文化会館4階で開かれる。FMシリーズのソフトおよび周辺機器などを取り扱っている約40社が出展の予定。本誌Oh! FMも出展する予定です。ぜひご来場ください。

なお，本誌ブースでこんなものをやってほしい，などご要望がありましたら，当編集部あてに葉書などでお寄せください。

Information

▶このコーナーのために幅広いご意見・ご感想・ご要望などをお寄せください。内容は問いません。イラストなども歓迎です。アンケートハガキに書き足らない場合は，官製ハガキか封書を利用してください。掲載者全員にOh! FMステッカーをさしあげます。あて先は

〒102 千代田区四番町2－1
日本ソフトバンクFMエリア係

▶第2号の「**アニマルミステリー**」の正解は

●キーポイント2
……1400で定数 LIMIT （500でも可）より大きいか………大きい場合には変数 X に選ばれた動物の番号が記憶される。

●キーポイント3
行番号 1390 で PF 2 キーが再び押されたと判断したら……。

でした。正解者の中から抽選で
高松市・池田純子 流山市・小石川光範
児島郡・福島勇 町田市・甲斐哲也様ほか
46名の方にOh! FM Tシャツをお送りいたしました。

コピーサービスのお知らせ

▶創刊号品切れのためご迷惑をおかけしましたが，コピーサービスを始めます▶コピー代はA4版1枚につき20円，郵送料は実費です▶表を見てコピー代と郵送料を計算し，現金書留，為替，切手，いずれかの方法でFMコピーサービス係までお申しこみください▶表以外の記事のコピーは行っておりません▶差額がでた場合，切手で返させていただきます▶住所，氏名，必要な記事名をはっきりお書きください。

▶第4号は7月18日（月）発売予定です。

記事別コピー代

記事	ページ数	コピー代
新機種FM-7の全容と周辺機器	5	100
V3.0のニューコマンド	8	160
FM-7ハード解析	4	80
OS-09+BASIC-09	9	180
I/O拡張ユニット REX-8	3	60
構造図作成用 EASY-DRAW	5	100
FM-7ベンチマークテスト	2	40
FM-8用 点検・ワープロソフト	5	100
マシン語入門（前編）	10	200
F-BASIC中級入門（第1回）	4	80
ソフトウェア設計法入門（第1回）	8	160
FM-8 BIOS解説書	15	300
実用ソフト 金種計算プログラム	5	100
カードテレパシー	4	80

郵便料金表

重さ	ページ数	料金
～50g	1～8P	120円
～100g	9～21P	170円
～250g	22～58P	240円
～500g	59～	350円

定形外封筒の重さを含め済

▶第2号は，若干在庫がありますので，ご入用の方はお近くの書店にお申し込みください。（主な内容：FM-11の全容/高速3Dグラフィックス/各社FDDUの比較検討など）

Question & Answer

▶多色表示について…

Q FM-7の記事やカタログには，8色カラーグラフィックと書いてありますが，⑦のシステム解説のマニュアルのp.7-8，9には「LINE文を使ってディスプレイ上に64色の色を出力するプログラムです」「スクリーン命令等を追加し，64色の色が順次変わるプログラムです」などと書いてあります。FM-7でも64色まで色を出すことができるのでしょうか。（川口市・金丸豊樹・13歳）

A 人が色を認識するというのはどういうことでしょうか。世の中にはあらゆる色があり，無限といってもいいほどです。しかし，あらゆる物が結局いくつかの原子核や電子などでできているのと同様に，ありとあらゆる色も結局三つの色，赤，緑，青で表すことができます（光の三原色。これに対し色の三原色は赤，青，黄）。

人間の目は，現在の機械など足元にも及ばないすばらしい性能を持っていますが，ある程度以上の細かいものを見ることはできません。テレビの画面を虫めがねで見ると，点の集まりであることがわかると思いますが，我々の目の性能の低さに助けられ（？）てテレビを楽しむことができます。

FMシリーズの画面表示は640×200（FM-11は640×400も可）ドットで，各ピクセル（ドットのことをこう呼びます）ごとに8色指定できます。

横が640ドットもの分割ができるために人間の目の性能では隣り合う2本の線を分割して認識できず，ちょうどテレビが点の集合によりできているのと同様に，線を組み合わせて色を合成することができます。

よく肌色とか中間色を表示しているソフトウェアがありますが，交互に色を変えた線によって合成しています。

PC-8801などでは，タイルペイントと言い，配列の中身によりペイントする命令があり，中間色を表示しやすくなっていますが，F-BASIC（11を除く）にはこの便利な命令がないために，しかたなくLINE命令により中間色を作っています。

なお，きれいな中間色を出すためには，普通のテレビでは無理が多く，中解像度以上のディスプレイが必要です。

▶エディタについて…

Q 第2号に載っていたウエディングレースをFM-7で入れて，プログラムの修正をしているときです。誤ってGO.35と入れてしまい，GO.3500と直したのですが，LISTをとると直っていませんでした。また，他のプログラムで色の指定をしたとき，1～5の指定はきちんとできるのに，6にすると白になり，7でも白になりました。このようなトラブルはパソコンでは，よく起こるのでしょうか。（松山市・福原正靖）

A FMシリーズのBASICでサポートされているエディタは，フルスクリーンエディタといい，他の機種にはない特長の一つで，慣れると非常に便利なものです。

スクリーン上の任意の点を複数修正しても修正可能ですが，内部の処理の都合でAUTO命令で出力したラインについては修正を受け付けてくれません（11を除く）。

しかも，エディタはそのことに関してメッセージを出してくれず，しっかり修正したつもりでも変化していないことがあります。このようなことがないようにするには，必ず修正するときは一度LISTをとり，そのうえでエディットを行えばよいのです。

また，よくやる間違いに図1のようにカーソル移動キーを使い，思ったとおりのエディットができないというのがあります。

カーソル移動キーのスクリーンに対する処理はただカーソルを動かすのみで，プログラムの内容に変化を与えません。つまりカーソルを使ってスクリーンの上につくった空白は内部に収納されません。図1のような場合はカーソル移動キーではなく，スペースバーで空白を作ってください。

図 1

```
10 PRINT "          A"
        カーソルで移動
          ⇩
10 PRINT "A"
```

また，カーソル移動キーなどのコードを一般にコントロールコードといい，表示はされませんが処理の対象にはなります。これらのコードを入力するには，左側にある CTRL キーを押しながら英字のキーを押せばよいのです。

実行時にこれらのコードを出力する場合は，

```
PRINT␣CHR$ (コントロールコード)
```

とすればよく，例えばカーソルを上に移動したければ，次のようにします。

```
PRINT␣CHR$ (&H1E)
```

KEY命令を使って，これらのカーソルコントロールキーなどをファンクションキーに登録したいときも同様にキャラクタコード表（マニュアルの付録参照）を使います。例えばFILES "CASØ:⏎をPF10に登録するときは，次のようにします。

```
KEY␣10, "FILES" +CHR$
    (&H22) + "CASØ:" +CHR
    $(&HD)
```

また，二つ目の質問は，前に入れたプログラムでCOLOR=（6,7）が実行されていたためで，新たなプログラムを入力する際には，リセットをかけるか，直接モードで

```
FOR I=Ø TO 7:COLOR=
    (I,I):NEXT
```

を実行してください。この理由についてはマニュアルのCOLORコマンド形式3をよく読んでください。

＊　＊　＊

FMに関する質問をお寄せください。質問内容は文書で，できるだけ具体的にお書きください。ソフト・ハードなど幅広い質問をお待ちしています。ただし，マニュアルを読めばすぐわかるような質問はご遠慮ください。**住所・氏名・電話番号**などを明記のうえ，**Oh!FM編集部Q&A係**あてお送りください。

料金受取人払

郵便はがき

102-

麹町局
承認
245

差出有効期間
昭和59年4月
19日まで

(受取人)

東京都千代田区四番町二―一

株式会社　日本ソフトバンク

出版部行

愛読者アンケート

読者の皆さまのご意見をお寄せください。誌面づくりに反映させるための貴重な資料となりますので、ご協力ください。

○本誌をご覧になった感想

○いちばん良かった記事

○興味のなかった記事

○載せてほしい記事・内容を具体的に

○パソコンをお持ちですか

ある（機種名　　　　購入　　年　　月）　ない

（オプション類<漢字ROMなど>　　　　）

（周辺機器　　　　メーカー名　　　　）

ある場合その用途を具体的に

ビジネス用

パーソナル用

○Oh！FM以外に読んでいる雑誌

FM　第3号

氏名	年齢	性別	パソコン歴 年　か月
住所 〒 Tel			
職業(勤務先，部署，役職) ※			

※学生の場合，学校学部学年を記入してください。

ショートプログラム集

長ーいリスト打ちにはもう疲れた，短くて気のきいたプログラムはないか，という人のためのコーナーです。

第2号のReader's Areaで呼びかけたところ，多くの方々から応募作品が送られてきました。今号はその中から1点と，編集部のデモ作品1点を掲載いたします。

このショートプログラム集は随時受け付けておりますので，右記の応募要領により，ご応募ください。テクニックの効いた美しいプログラムをお待ちしています。

応募要領

- ■1行80字以内で，できるだけ30行に収めてください。
- ■きれいなプログラムを望みます。マルチステートメントは多用しないでください。
- ■プログラムは，カセットテープまたはディスケットにいれてお送りください。ディスケットの場合は，コピーしたのちお返し致します。
- ■プログラムの説明，使い方など必要なことを簡潔にまとめてください。
- ■プログラムおよび原稿は，多少修正させていただく場合があります。
- ■他誌との二重投稿，盗作など，モラルに反する応募は固くお断わり致します。
- ■応募原稿は原則としてお返し致しません。
- ■応募された方全員に，Oh！FMシールをお送り致します。
- ■すぐれた作品は本誌に掲載し，原稿料にかえてOh！FM特製Tシャツと図書券3,000円分をお送り致します。
- ■原稿にも，住所・氏名・年齢・職業などを明記して，〒102 東京都千代田区四番町2−1 ㈱日本ソフトバンク Oh！FMショートプログラム集係までお送りください。

ワードサーチ & リプレイス プログラム　FM-7, 8, 11

ディスクシステムで動作します。このプログラムは，プログラム中の任意の言葉を捜し，あるいは任意の言葉に置き換えるもので，プログラムのデバッグなどに役立ちます。

使用時には，まず変更するプログラムをディスクへアスキーセーブします。たとえば，

LIST"Ø:SOURCE"

などとしておけば結構です。ここでファイル名は任意です。

次に，このプログラムを走らせます。すると，ソースファイルを聞いてきますので，アスキーセーブしておいたファイル名を入力してください。

次にデストネーションファイルネームを聞いてきます。今度は変更後のプログラム名を打ち込みます。変更前と変更後のプログラム名を同じにしてはいけません。また，ワードを捜すだけなら入力の必要はありません。

さらにサーチワードを尋ねてきます。捜すワードを入力してください。言葉を書き換える場合は書き換える対象を入力することになります。

リプレイスワードの入力では，新たに書き込むワードを入力します。サーチだけする場合には何も入力せずリターンしてください。

変更した行，またはサーチの結果捜していたワードが発見された行は画面に表示されますが，プリンタに出力したい場合には行番号330のPRINT文を変更する必要があります。

```
100 DEFINT A-Z : TRUE=(0=0) : FALSE=NOT TRUE
110 ON ERROR GOTO 390
120 COLOR 5 : PRINT "WORD SEARCH & REPLACE": COLOR 7
130 LINE INPUT "Source file name : ";SOURCE$
140 LINE INPUT "Destination file name : ";DESTINATION$
150 LINE INPUT "Search word : ";SEARCH$
160 SEARCHLEN=LEN(SEARCH$)
170 LINE INPUT "Replace word : ";REPLACE$
180 IF SEARCH$=REPLACE$ THEN PRINT "Illegal strings" : BEEP : GOTO 150
190 IF REPLACE$="" THEN REPLACEFLAG=FALSE ELSE REPLACEFLAG=TRUE
200 :
210 OPEN "I",1,SOURCE$
220   IF REPLACEFLAG=TRUE THEN OPEN "O",2,DESTINATION$
230   WHILE NOT EOF(1)
240     REQUIRE=FALSE
250     LINE INPUT #1,CURRENT$
260     CP=INSTR(CURRENT$,SEARCH$)
270     IF CP<>0 THEN REQUIRE=TRUE
280     IF REPLACEFLAG=FALSE THEN 330
290       IF CP=0 THEN 320
300         CURRENT$=LEFT$(CURRENT$,CP-1)+REPLACE$+MID$(CURRENT$,CP+SEARCHLEN)
310         GOTO 260
320       PRINT #2,CURRENT$
330     IF REQUIRE=TRUE THEN PRINT CURRENT$
```

```
340   WEND
350 CLOSE
360 :
370 COLOR 5 : PRINT 'END OF EXECUTION': BEEP: COLOR 7
380 END
390 IF ERR=64 THEN KILL DESTINATION$ : RESUME
400 ON ERROR GOTO 0
```

Rainy Day　倍速FM-8, FM-7, 11

篠原 祐三(桶川市)

それは私の頭の中にまで雨音が入り込んできた日曜日でした。その前日，私は雑誌の記事を頼みに，わが愛機FM-8のサブCPUを倍速化させることに成功していました（それにしても，くやしいかな7・11…）。そんな前日のハンダごての興奮も手伝って作りあげたプログラムです。これといったテクニックは使っていませんが，間の取り方には気を使ってみました。

(編集部記)

このプログラムはタイミングなどFM-8の倍速やFM-7用に作ってありますが，普通のFM-8でも実行可能です。

これだけのプログラムで，あれだけの時間楽しませてくれるとは，作者のプログラムセンスに敬服させられます。画面の美しさも良好で，特に最後の画面は筆者の好みにぴったりで，非常に気に入ってしまいました。ただ，これだけの長さに押さえるために，マルチステートメントを多用しなければならなかったのか，見にくいプログラムになってしまったようです。とは言え，これだけの機能を持っていることから，今回選ばせていただきました。次回には，見やすくさらに楽しいプログラムを送っていただけたらと思います。なお，FM-11の方は，最初にSCREEN 0を実行してください。

```
10 '*** RAINY DAY  *** for FM-8 fast sub-CPU
20 DIM A%(210):COLOR7,0:WIDTH80,25:DATA2,6,4,5,1,3
30 GOSUB270:FORI=0TO2000:NEXT:COLOR0:GOSUB280:COLOR7
40 BEEP1:BEEP0:FORI=0TO20STEP5:CIRCLE(319,99),I,5:NEXT
50 GET@(299,80)-(319,99),A%,G,5
60 N=5:S=1500:GOSUB140:N=10:S=200:GOSUB140:N=500:S=1:GOSUB140
70 N=100:S=50:GOSUB140:N=400:S=1:GOSUB140:GOSUB180:N=40:GOSUB160:GOSUB180
80 FORI=0TO10:A=RND(1)*630:B=RND(1)*190:GOSUB110:FORL=40TO190STEP10
90 CIRCLE(A,B),L,7:CIRCLE(A,B),L-20,5:NEXTL:GOSUB130:C=RND(2)*2000
100 FORK=0TOC:NEXTK:NEXTI:GOTO190
110 PSET(A,B,7):BEEP1:PRESET(A,B):BEEP0
120 CIRCLE(A,B),20,7:CIRCLE(A,B),30,7:RETURN
130 CIRCLE(A,B),L-20,5:CIRCLE(A,B),L-10,5:RETURN
140 FORI=0TON:A=RND(1)*599:B=RND(1)*160:FORL=0TOS:NEXT:BEEP1:BEEP0
150 PUT@(A,B)-(A+40,B+39),A%,PSET,5:NEXT:RETURN
160 FORI=0TON:A=RND(1)*630:B=RND(1)*190
170 BEEP1:BEEP0:FORL=10TO50STEP10:CIRCLE(A,B),L,7:NEXT:NEXT:RETURN
180 BEEP1:BEEP0:LINE(0,0)-(639,199),PSET,5,BF:RETURN
190 FORI=0TO150:CONNECT(0,I)-(639,I),5
200 FORL=0TO5:X=RND*639:X1=RND*639:S=RND*5+I+2:CONNECT(X,S)-(639-X1,S),7:NEXT
210 CONNECT(0,I)-(639,I),1:NEXT:CONNECT(0,I-2)-(639,I-2),5
220 FORP=.99TO.52STEP-.016:RESTORE:K=0:FORL=1TO6:READCL:FORI=1TO6:K=K+1
230 CIRCLE(319,155),150+K,CL,,P-.016,P
240 CIRCLE(319,155),150+K,CL,.25,1-P,1.016-P:NEXT:NEXT:NEXT
250 PRINT@(410,20),&H4032,0,&H246C,0,&H243F,0,&H212A:LOCATE0,22
260 GOTO260
270 PRINT@(200,90),&H312B,0,&H244E,0,&H467C
280 PRINT@(300,130),&H243D,0,&H2437,0,&H2446,0,&H2145,0,&H2145:RETURN
```

漢字ROMのない方は次の3行を変更してください。

```
250 SYMBOL(410,20),"ハ レ タ !",2,2,7
270 SYMBOL(180,90),"ア メ ノ ヒ",2,2,7
280 SYMBOL(300,130),"ソ シ テ ・・・",2,2,7:RETURN
```

FM-7ハードウェア回路図と解説〔後〕

15

サブCRT RAM

VRAM用として16KバイトのDRAM（MB8116H）を24個用いてRED，GREEN，BLUEそれぞれ16Kバイトの容量を得ている。

M78，M79，M80，M81（MB74LS244）はサブCPUのデータバスへ接続するためのデータバスバッファである。M78がデータの書き込み用，M79～M81がデータの読み込み用である。

SVRADRSはCRTコントロール用LSIであるM124（MB60H010）より出力されるVRAMアドレスだ。SVDATAB, SVDATAR, SVDATAGはCRTへ出力されるビデオデータである。

16

サブCRTインタフェース

M68，M69，M70（SN74LS166）を用いてCRTへ出力するRED，GREEN，BLUEそれぞれのビデオデータを，パラレルからシリアルへ変換する。

M49（MB15021）は500ゲートのバイポラゲートアレイであり，パレットレジスタの機能を持たせてある。

17

キーインタフェース

M125（MB88401）はNMOS 4ビットワンチップマイクロコンピュータで，キーボードのスキャンに用いる。コネクタ（CN6）を通じてキーマトリクスに接続される。

M113（MB74LS244）はサブCPUのキー入力ポートとして，M88（MB74LS244）はメインCPUのキー入力ポートとして用いられる。

M77（SN74LS273）は$FD02番地の出力ポートで，IRQ割り込みのマスクに使用される。

18

拡張バスバッファ

本体背面の50ピンの拡張バスポートおよび本体上面の34ピンのオプションカードスロットへ出力するためのバッファ回路である。

データバス，アドレスバス，制御信号はバッファリングをしてからそれぞれのコネクタに接続されている。

22

メインPSG

M7（AY-3-8913）はPSG（Programable Sound Generater）と呼ばれるサウンドコントロール用のICである。PSGの出力はM8（LM386）によるオーディオアンプで増幅されてからスピーカへ出力される。

M32（MB74LS139）はPSGに対するアドレスデコードに，M21（MB74LS74A）は$FD00番地の出力ポートとして用いられている。bit 0をPSGのBC 1端子に，bit 1をPSGのBDIR端子に接続して，メインCPUのPSGに対するデータのコントロールを行っている。

★耳よりな情報★

「マイコン利用者認定試験」受験受付はじまる

日本マイコンクラブは，本年度より「マイクロコンピュータ利用者認定試験」を実施，その第1回試験が7月24日(日)行われる。

同試験は，マイコンに関する基礎知識からシステム運用段階に至る応用能力などを客観的に認定し，マイコンの利用者や指導者などの資質，利用能力の向上を図ることを目的としている。したがって，通商産業省が実施している「情報処理技術者試験」のように資格や免許を与えるものではなく，一定の能力水準に達しているという認定証を発行するに留まる。

クラスは1級から4級の4区分があり，4級から順次合格しないと，上級クラスは受験できない。ただし，4級，3級は同日中に受験できる。

このため本年度の試験は3，4級だけとなるが，来年度2級を受験したい人は，ぜひ今回3級まで受験されたい。

札幌，仙台，東京，長野，名古屋，大阪，広島，松山，北九州，福岡，熊本の11会場が予定されている。受験願書の受付は5月10日～6月10日。詳しくは，〒105 東京都港区芝公園3-5-8 機械振興会館内　日本マイコンクラブ認定試験係　☎03(438)1869まで。

⑮図 サブCRT RAM

⑰図　FM-7キーインタフェース

⑱図　FM-7拡張バスバッファ

⑲図 FM-7コネクタ(拡張, Z80)

⑳図 FM-7コネクタ(PSG)

㉒図　FM-7メインPSG

ビジネスマン・OLのための本格派教室です。

ウチダマイコンスクール

あなたは「できない」

今…
OLからELへ変身!

人事部長殿!

札幌岩崎教　☎011(221)6161

横浜西口教室　☎045(314)6201

伊東教室　☎0557(36)8500

浜松駅南教室　☎0534(53)3151

金沢教室　☎0762(47)5107

長野教室　☎0262(28)7088

千種教室　☎052(722)0687

八事教室　☎052(834)0355

岐阜教室　☎0582(73)5891

津教室　☎0592(25)6251

松阪教室　☎0598(51)4312

京都・伏見稲荷教室　☎075(641)1706

梅田教室　☎06(345)2111

日本橋教室　☎06(641)4188

堺東教室　☎0722(23)5266

和歌山教室　☎0734(32)3779

兵庫教室　☎078(577)3435

ビジネスマン・OLのための本格派教室です。

と逃げるのか「やってみよう」と乗り出すのか？

いま“マイコン武装”を

●いま、日本のビジネス社会では、文字通り「秒単位」のスピードでマイコン革命が進行しています。マイコンの正しい知識と正確な技術を伴った“マイコン武装”は、運転免許と同様、これからのビジネスマン、OLにとって絶対に欠かすことができません。この現実から「逃げる」のか、それとも「挑戦する」のか——その意志決定こそ、あなたがマイコン時代に生き残れるかどうかのバロメーターになるのです。

全国どこの教室でもハイ・レベルな均一カリキュラムを

●ウチダマイコンスクールは、全国どこの教室でも質の高い均一なカリキュラムを受講できます。特に、企業では今春入社のフレッシュマンへの社員教育にご活用いただければ、さらに効果的です。

社員教育の一環に「いつでも」「どこでも」………一度ご相談下さい。

会津若松教室 ☎0242(22)4441　浦和教室 ☎0488(31)3655　池袋教室 ☎03(984)1051　銀座教室 ☎03(541)1481

泉大津教室 ☎0725(23)0610

四日市教室 ☎0593(51)1651

大阪本町教室 ☎06(261)5791

佐世保教室 ☎0956(24)3398

本格的業務処理用
パッケージ・プログラムを用意しています。

■**経理業務パッケージ**　(200,000円)
複式簿記を基本として、仕訳入力から決算処理まで、日常経理業務全般。

■**給与計算パッケージ**　(200,000円)
従業員300名までの事務所用、月例給与、賞与、年末調整、社会保険算定等。

■**販売・仕入・在庫パッケージ**　(200,000円)
仕入・販売から在庫管理まで有機的に結びつけ日常業務から棚卸処理まで。

■**ファイリング・パッケージ**　(70,000円)
各種情報(顧客・文献・不動産・資産・貸借物件等)の記録、管理。

■**プログラミング支援パッケージ**

●**統計計算サブルーチン・パッケージ**　(70,000円)
各種統計処理の計算処理部分をサブルーチンとして活用できるよう用意。

●**システム初期設定パッケージ**　(10,000円)
システム起動時からジョブメニューへの移行を自動化し、扱いを簡素化する。

●**文字パターンパッケージ**　(10,000円)
システムが用意している文字のパターンを拡大表示、新規作成、パターン変更登録できる。

●**逆アッセンブラ**　(4,000円)　●**メモリー・ダンプ**　(4,000円)

ウチダマイコンスクール

本部：〒104 東京都中央区銀座4丁目9番13号 国土館ビル
☎03-541-1481(代)

株式会社 内田洋行

FM-7、FM-8、100

IRIS80 基本カード

FM-7、FM-8の
情報管理の

タイトル
年齢順による並べかえ
合計、平均、最大
プリントアウトキー
単位

IRIS80（アイリス80）は、情報化社会のニーズから生まれた電子カード式情報管理用ソフトウェア。自由な項目設定で、顧客管理、成績管理、人事管理、問合せ管理、不動産情報、文献検索、在庫管理、業務管理、名刺管理など、様々な情報管理に利用できます。プログラミングは全く不要。ディスプレイを見ながら、一覧表の作成、項目間・カード間の計算、並べ替え、宛名ラベルの作成が思いのままにできます。応用カードプログラム、PaPaシリーズと組み合わせれば、いっそう実務に合った情報管理システムが実現します。

●基本カード

ファイル名や日付などの欄と最大20項目まで記入できる項目欄と、そのデータ記入欄で構成されている基本カードのフォーマットが、パソコンのディスプレイ上に映し出されます。

●データ編集

完全画面編集でカーソルの移動によりデータを自由に編集。画面に写っているそのままがフロッピーディスクに記録される、つまり画面が1枚のカードとなるわけです。

●検索、並べ替え、サブファイル

条件検索では、20項目すべてに重ねて条件を指定したり、並べ替えを同時に行なえます。さらに検索、並べ替えをしながら、必要な項目だけを選んで新しいファイルを作成するサブファイル機能があります。

●レポート

検索条件、並べ替えを指示しながら、任意の項目に対して順序を自由に指定し、一覧表を作成することが可能。またプリンタの出力では、プリンタに合わせて30文字から255文字まで指定することができます。

●項目間の計算

画面に表示された項目の間の演算（四則演算・関数演算・文字演算）ができます。また、新しい項目に計算式を記入して、その計算結果をデータとして保存することも可能。

●ラベルの打出し

検索条件を与え、必要なカードから住所や氏名だけをラベルに打ち出せます。ダイレクトメール用に最適です。

PCA 誰にでも手軽に使える 高級ソフト

使いやすさとトータル・パフォーマンスを実現しました。

PCA高級財務会計システム

◆富士通FMシリーズ用◆

〈監修〉 公認会計士　吉村　成弘
ピーシーエー株式会社取締役

〈開発・販売〉 ピーシーエー株式会社
都築電産株式会社 TEL(03)433-2171

〈システムの価格〉
A(元帳、試算表、決算書)　¥ 98,000
B(A＋補助簿、手形管理)　¥148,000
C(B＋部門管理、その他分析)　¥198,000

〈特　長〉
○あなたの会社に合わせた科目登録や補助簿登録ができ、合計金額1,000億円未満の会計処理ができます。
○主要帳票はオフコン並の専用用紙が用意されておりますので、銀行や税務署等にそのまま提出できます。
○主勘定科目最大240、補助科目最大240が用意され、補助科目の活用により、売掛金、買掛金、手形管理など広範囲な処理ができます。
○部門設定が９分類でそれぞれの損益計算書ができます。
○１ヶ月 2,000仕訳の処理ができます。
○入力方法は入金、出金、単一振替、複合振替が可能で、入出金は自動仕訳方式になっています。
○画面は漢字の大文字で表示され、非常に見やすくなっています。
○日常良く使われる主要48科目は、ワンタッチ入力方式になっています。(下記キーボード参照)
○帳票名を選択するだけで自動的に分類集計する等、使いやすく、正確で、しかも主要部分は機械語処理により、トータル・パフォーマンスを追求しています。
○処理範囲によりA.B.Cと３段階のシステムが用意され、必要なレベルを選択できます。

MENU 1

```
0: MENU 2へ              6: 科目・残高の登録・変更
1: データ 入力            7: 日 計 表
2: データ 検索・訂正       8: MENU 3へ
3: チェックリスト
4: 仕 訳 帳
5: 科目残高登録リスト
```

処理を 選択して下さい ?

《システムの概要》

	項目	A	B	C
日常処理	前準備処理	○	○	○
	データ入力作業	○	○	○
	データの各種検索，訂正	○	○	○
	入力データーチェックリスト	○	○	○
	仕訳日記帳	○	○	○
	日計表		○	○
任意処理	元帳	○	○	○
	補助元帳		○	○
	手形管理表		○	○
	補助科目残高一覧表		○	○
	合計残高試算表(B/S, P/L)	○	○	○
決算処理	決算報告書			
	貸借対照表，損益計算書	○	○	○
	販売管理費明細書	○	○	○
	製造原価報告書	○	○	○
	利益処分計算書	○	○	○
その他管理資料	取引先別総合収支明細表			○
	資金繰実績表			○
	経営分析表			○
	部門別損益計算書			○
	部門別P/L一覧表			○
	月次残高推移表			○
	三期連続P/L対比棒グラフ			○

※税務申告ソフト開発中！

FM-8&FM-7 Disk Utility
HOSPITAL

健康診断

ディスケット上に機械的なキズ(Physical Error)がないかどうか調べます。全トラックについて読みだしテストを行ない、必要に応じてFormat・システムコピー等をする事が出来ます。

胃腸科

F-Basic用ディスクのFAT/Direetryを解読します。FAT/Direetryの指定のほかに"FILES"の指定をすると、Killされたファイルも含めて、ファイル名・クラスタ使用状況の一覧表が示されます。

X線診断

FM-8仕様に限らず、両倍(40T)・片倍(35T)ディスケットの任意のセクタがスクリーン上に示されます。他社仕様のディスクを読んで理解するためには、その社の呼称法、セクターの読み出し順序の法則を、F-Basicでの法則に換算しながら進めて下さい。

整形外科

上と同じく、任意のセクタをバイト単位でスクリーン上で修正できます。このルーチンも無意味なトラックを指定しない限り他社仕様のディスクも対象となります。間違って入力しても(CR)しない限り何度でも修正出来ます。

脳神経外科

プログラムの収納の構造が調べられ、ツブれたプログラムの修復・他社仕様のディスケットの移植の準備に使われます。各クラスタの最初のセクタを表示します。また、このセクタの前半分と直前のクラスタの最後の半セクタを表示することや、直前のセクタのつながりを見たり、プリンタがあればCopyを取ったり、新しいクラスタの指示も行なう事が出来ます。

価格
15,000円
〒300

産婦人科

F-Basic用のFormat・システムコピー・初期化とVolcopy等を実行します。Format・Volcopyは、他機種、他言語にも使えます。いずれの場合も、＃0上のプログラムディスケットが読み込まれ実行が始まります。＃0に入っているプログラムディスケットを、出来るだけ出し入れしないようにしていますのでFujitsuから提供されるものよりは作業が簡略化されています。

人工受精

片倍(35T)ディスケットの任意の面の全Copyに使います。表面(例えばPC仕様の片面)、裏面(例えばIf-800)が指定できます。片倍(35T)のディスケットは普通反対面が、Format していないので、通常のVolcopyが出来ません。この様な場合、表あるいは裏を指定して、その面を35トラックだけVolcopyするルーチンです。尚、PASOPIAのように両面で35Tの場合は、このルーチンを表裏2度走らせば良いのです。このプログラムでも、産婦人科でもVolcopyは、Killされたファイルも含めて、内容を判断する事なく、完全Copyします。

心臓移植

PC-8801(NEC)の両倍仕様の、任意の一つのプログラム・ファイルをFM-8仕様にCopyします。それに先立ってPC-8801仕様のディスケットの構造を、調べる事が出来ます。このルーチンのうちファイルコピープログラムそのものは、オリジナルがAscii Saveされていなくても動きますが、目的が移植にあるわけですから、オリジナルのAscii Saveは必項です。このプログラムではドライブ番号、トラック番号等をいきなりFM-形で書かないで、一度PC(両面)形で書いて改めて換算するようにしています。また、片面倍密度仕様のディスケットでも使用出来ます。

健康手帳『ディスクインウィークス』自費出版発売中!(2,000円)

FM←→PCディスクコンバージョンユーティリティ『DISK-MUCH』お求めのユーザーの方々には、手数料6,000円で交換します。オリジナルマスターをご返送下さい。

不思議…?

OUT IN

テープ・ロード・アダプター

テレコとマイコンの間に接続して、波形を成型するので、LOADミスがほとんど、生じなくなります。

価格 8,800円(〒200)

WORLD WIDE COMPUTER SUPER SHOP
COSMOS™ 岡山

〒700 岡山市南方5-6-5 今田ビル2F(県営グランド入口向)
TEL(0862)54-7474 ミニファクス(0862)54-7481
年中無休 AM10:00~PM7:00
通販は現金書留又は郵便振替 岡山4-12524 コスモス岡山

簡易言語
NEW VIP付

FUJITSU FM-7

新しい感性を伝えてくれるFM-7 限定100台 即納

¥21,000分プレゼント 本体＋プリンター＋フロッピーディスク以上のセットをご購入の方に。

● ¥3,000分のプリンター用紙1000枚。(本体とプリンターをご購入の方に)
● ¥18,000分のブランクディスケット10枚。(本体とフロッピーディスクをご購入の方に)

特別高額下取制度	
下取機種	下取差額
PC-8001	¥ 80,000
MZ-80B	¥ 55,000
MZ-1200	¥ 90,000

毎月わずか¥3,000

●ご注文セットNo.11 FM-7

入門セット

本体＋ニデコ14″高解像度カラーディスプレイ＋FM-データレコーダ

No.1770	MB25010	¥126,000
No.2062	NH-14DD(ケーブル付)	¥ 69,800
No.1778	MB27501	¥ 12,800
合計標準価格		~~¥208,600~~

【大特価クレジット】

① ¥3,000×24回 ㊙2万 ㊍3.4万×4回
② ¥5,740×36回 頭3万 ボなし
③ ¥5,000×24回 頭なし ボ2.8万×4回
④ ¥6,760×36回 頭なし ボなし

毎月わずか¥3,000

●ご注文セットNo.12 FM-7

お買得セット

本体＋漢字ROMカード＋ニデコ14″高解像度カラーディスプレイ＋FUJITSU1ドライブミニフロッピーディスク＋EPSONプリンター

No.1770	MB25010	¥126,000
No.1771	MB22405	¥ 35,000
No.2062	NH-14DD(ケーブル付)	¥ 69,800
No.1784	MB27607＋(22407、SMO7317-F121)	¥119,800
No.1758	RP-80＋(ケーブル)	¥ 96,000
合計標準価格		~~¥446,600~~

【大特価クレジット】

① ¥3,000×48回 頭5万 ボ4.4万×8回
② ¥8,590×60回 頭5万 ボなし
③ ¥10,000×24回 頭なし ボ6.5万×4回
④ ¥14,570×36回 頭なし ボなし

毎月わずか¥5,000

●ご注文セットNo.13 FM-7

お買得セット

本体＋漢字ROMカード＋ニデコ14″高解像度カラーディスプレイ＋EPSON2ドライブミニフロッピーディスク＋SEIKOSHA漢字プリンター

No.1770	MB25010	¥126,000
No.1771	MB22405	¥ 35,000
No.2062	NH-14DD(ケーブル付)	¥ 69,800
No.2036	Disk-80FMK II＋(CBL-3、disk-I OF、SMO7317-F121)	¥155,000
No.2063	GP-550E＋(ケーブル)	¥127,300
合計標準価格		~~¥513,100~~

【大特価クレジット】

① ¥5,000×60回 頭5万 ボ2.8万×10回
② ¥8,580×60回 頭10万 ボなし
③ ¥5,000×48回 頭なし ボ4.8万×8回
④ ¥16,250×36回 頭なし ボなし

サンシャインビル
57F 展望ショップ (地上228m)

——キャットジャパン・マイコンプラザ直営店——

サンシャインビル・57F展望ショップ
営業時間A.M.10:30～P.M.7:00 TEL03-988-1125
(ショップのみ水曜定休・電話注文は年中無休)

5月18日より全国一斉受付開始

●印の25ヶ所は24時間受付、他は9:30～20:00受付、いずれも年中無休。

北海道地区 ●札幌(011)644-0375 **東北地区** 盛岡(0196)53-5371 ●仙台(0222)21-3811 **関東地区** 茨城(0292)26-5575 ●宇都宮(0286)37-1977 ●高崎(0273)22-8211 ●大宮(0486)44-0521 ●浦和(0488)87-9521 春日部(0487)36-3861 所沢(0429)26-7335 ●千葉(0472)25-2028 ●船橋(0474)67-9115 ●柏(0471)55-2218 ●横浜(045)712-0402 **東京地区** ●池袋(03)983-1369 神田(03)861-5700 ●新宿(03)375-1861 ●町田(0427)29-5731 調布(0424)88-9421 ●八王子(0426)26-7701 **東海地区** ●静岡(0542)58-6611 **中部地区** ●長野(0262)43-7812 **北陸地区** ●新潟(0252)31-6398 金沢(0762)22-7011 **中京地区** ●名古屋(052)264-4651 岐阜(0582)66-5917 ●京都(075)255-4637 津(0592)26-1601 奈良(0742)22-1048 **阪神地区** ●大阪(06)365-1706 ●神戸(078)577-7728 **山陽地区** ●広島(082)294-6402 ●岡山(0862)25-2881 **四国地区** ●松山(0899)52-7600 **九州地区** 北九州(093)522-5346 ●福岡(092)473-6690 鹿児島(0992)57-6388 **本社受付本部 03-983-1333**

■すでにご注文いただいております商品のお届け時期(納期)や、メインテナンスその他のお問い合わせは下記のテレフォンサービスセンターへお電話ください。
●札幌(011)611-8481 ●仙台(0222)63-4964 ●東京(03)983-1412 ●名古屋(052)264-9543 ●大阪(06)365-1705 ●広島(082)292-1380 ●福岡(092)473-5413

キャットジャパンリミテッド株式会社 本社・〒170 東京都豊島区池袋サンシャイン60 24F

お支払い方法……クレジットの月々のお支払いは①銀行口座のある方は、自動引落。②銀行口座のない方は、お近くの都市銀行・地方銀行・信用金庫・信用組合・農協等の金融機関(郵便局の場合は郵便振込)よりクレジット会社宛にご送金いただきます。

大特価クレジット 月々のお支払いを魅力のコースで。	**低金利クレジット** 頭金なしで60回までOK。
24時間受付 夜型の方でも好きな時にTELできる。	**日曜配達可** 不在がちな方、日曜なら大安心。
頭金なし 電話一本でOK。らくらくクレジットです。	**無料配送** 全国どこでも配送料はすべて無料です。
製品先取り 電話一本、手続き完了！製品即納。	**完全保証** 製品はすべてメーカーの完全保証付です。
支払い約2ヵ月後 お支払いは、のんびりと。	**カレッジクレジット** 18歳以上の学生の方、保証人不要。
高額下取制度 高額下取りでラクラク買い換え。	**クレジット自由自在** お支払い回数は1～60回までOK。

光栄マイコンシステム★

バラエティーソフト

ウォーゲームシリーズ

■信長の野望 ￥4,500

全国各地を猛将が割拠する戦国時代彗星の如く出現し、天下統一という野望を胸に戦さを挑む、戦国の革命児、織田信長。彼の人生は戦さの連続だ！今、貴方は「信長の野望」を果たせるか！●機種／PC-8801●種類／テープ。

ロールプレイングゲーム

■クフ王の秘密 ￥4,500

エジプト最大を誇るクフ王のピラミッド。ここには莫大な財宝が隠されている。もしこの財宝を探し当てることが出来たら、たちまち大富豪だ！だが、今貴方の手元には手掛かりとなるものは何もない。さぁ、この財宝さがしに貴方はどう挑むか！●機種／PC-8001●種類／テープ。

アプリケーション

■ナイトライフ ￥4,500

これほど実用的で家庭に密着したソフトはない。その日の体調にあったフォルム、同時に安全日もお教え致します。豊かなナイトライフをお過し下さい。（結婚祝いにもどうぞ）●機種／PC-8001 PC-8801、FM-7・8●種数／テープ（ディスクは開発中です）。

君の頭脳とマイコンの

決戦 シミュレーションゲームシリーズ

■投資ゲーム ￥5,000

ソ連大凶作、金放出か。中東に戦火、ウォール街の動きに東京市場は……？貴方の分析はいかに。●機種／PC-8001、PC-8801、FM-7・8、MZ-80B（HuG-BASICとグラフィックラム1必要）、レベルIII●種類／テープ。

■地底探検 ￥4,500

たび重なる危機、おそいかかる怪獣、地底人なぎ倒し財宝を手に入れる事が出来る……。●機種／PC-8001、PC-8801、FM-7・8、レベルIII●種類／テープ。

戦略戦術を

競う ウォーゲームシリーズ

■バルジ大作戦 ￥4,500

ドイツ軍が命運を賭けた最後の決戦。ヘクス画面が前後左右にスグロール、スゴイ!!（グラフィックラム1必要）●機種／MZ-80B●種類／テープ。

■川中島の合戦 ￥3,500

貴方は武田信玄。騎馬・弓・歩兵他の部隊を総指揮して、大挙して押し寄せる上杉勢に勝てるか。●機種／PC-8001、PC-8801、FM-7・8、MZ-80B（HuG-BASICとグラフィックラム1必要）、レベルIII●種類／テープ。

■コンバット ￥4,500

貴方はサンダース軍曹。カービー、ケリー、リトルジョン達を率いて独軍陣丘の上のトーチカを攻略。●機種／PC-8001、PC-8801、FM-7・8、レベルIII、MZ-2000（グラフィックラム1必要）●種類／テープ。

■ノルマンディー上陸作戦 ￥4,500

1944年Dディ。ノルマンディーより上陸し、ベルリンに向い大進撃を開始した連合軍を貴方（独軍参謀長）は食い止められるか。●機種／PC-8001、PC-8801、FM-7・8、MZ-80B、MZ-2000（グラフィックラム1必要）●種類／テープ。
※ボード別売…………￥1,200＋送料（￥300）

■ダス・ブート ￥3,500

Uボート艦長である貴方の使命は輸送船団に壊滅的打撃を与える事だ。敵護衛艦との攻防はいかに！（グラフィック機能豊富）●機種／PC-8001、PC-8801、FM-7・8、MZ-80B（HuG-BASICとグラフィックラム1必要）、レベルIII●種類／テープ。

■空中戦 ￥3,500

貴方は海軍航空隊の編隊長。太平洋上で遭遇した敵戦闘機編隊との死闘が、いまさに始まろうとしている。●機種／FM-7・8、PC-8801●種類／テープ。

史上初 新ジャンルゲームソフト ファンタジーアドベンチャー

■ドラゴン&プリンセス ￥4,500

王国より選び抜かれし戦士よ、今こそ城の宝を取りもどせ、報酬は姫と金貨にあり。●機種／PC-8001、PC-8801、レベルIII、FM-7・8●種類／テープ。

SFシミュレーション

■サンセット・イン・ラディック ￥4,500

敵の基地に忍びこみ、機密文書を盗み出せ、これが貴方にあたえられた指令だ。G型バトル・ドレスを身にまとい、今戦いか始まる……。●機種／PC-8001、その他開発中●種類／テープ。

パロディシミュレーション

■ホイホイ ￥4,500

●機種／PC-8801●種類／テープ。

近日発売！

日本古来の知的ゲーム

■連珠 ￥3,500

別名5目ならべとして親しまれ、15目の盤上に戦われる高度なテクニック。さぁ、貴方はこの勝負はどう挑むか……。●機種／PC-8001●種類／テープ。

驚異 ゲームパックシリーズ

PC-8001、PC-8801用各￥3,800
（PC-8801用はSilver Pack IIのみ）

■Silver Pack II

●モグラたたき●スターキャッチャー●スタートレック●テニス●スネーククネクネ●イーブン●フィーガーショック●さいころ賭博●クラッシュ●ACYデュシー●リバース。

■Golden Pack II

●ミグ25●藤原京エイリアン●ラビットハンター●15ゲーム●ザ・プリズナ●センチピード●2次元迷路●ハノイの塔●ルーレット●恋うらない●ビット&ブロー。

■Bronze Pack 6

●アレンジボール●国盗ゲーム●三次元迷路●バドルオブタンク●スペースチェス。

■Platina 5

●チェッカー●ディグ・ダウン●ニム●山くずし●ゴルフ。

■いろは ●PC-8001用日本語ワードプロセッサー

●マシン語による高速処理●漢字ROM不要●漢字プリンター不要●漢字タブレット板不要●本体解造不要

特徴 ▶仮名漢字変換方式▶熟語変換可能（辞書部オプション）▶拡大文字、アンダーラインプリント可能▶カーソルエディット方式による豊富なエディター（挿入、削除訂正、右よせ、センターリング他）▶JIS第一水準漢字を内蔵▶片仮名、平仮名、英数記号、ギリシャ文字、ロシア文字その他を内蔵▶外字登録可能▶美しいプリンター出力（16×16ドット構成で普通文字文）▶縦書き出力可能（オプション）

驚異の低価格

●基本プログラム…………￥16,800
●熟語辞書(オプション)……￥9,800
●縦書き機能(オプション)…￥4,500

使用可能プリンター／PC8023、PC8821、PC8822、MP80II、MP83、RP-80、GP250F。

●お知らせ●
光栄バラエティーソフトをお買上げいただき、誠に有難うございます。当社ソフトをお求めになりたい皆様は、本社宛現金書留でお送り下さい。最優先で皆様のお手元にお届け致します。尚、本社からの送料はサービスさせていただきます。
（株）光栄

お求めは全国マイコンショップ又は本社あてに現金書留（送料サービス）に機種を書いてお送り下さい。

★プログラムをお売りになりたい方、本社・襟川までご連絡下さい。★社員募集：マイコン、パソコンに興味をお持ちの方、貴方の才能を当社で生かしませんか！

特注ソフト・各社パソコン取扱い
秋葉原・生協に負けない実力価格。

光栄マイコンシステム★

本社：〒326 足利市通1丁目2677 ☎0284-41-5911代 日・祭日休
足利店：〒326 足利市通1丁目2677 ☎0284-44-1581 (11:00〜20:00)年中無休
日吉店：〒223 横浜市港北区日吉本町1876 ☎044-61-6861 (12:00〜20:00)日・祭日休

ザ・パッケージ シリーズは、さらに充実します。

ザ・パッケージシリーズカセット版は誰でも簡単に使え、しかも実務に役立つことが基本設計です。

（全プログラム共マニュアル付）

プログラム名	仕様概要	標準価格	機種
実戦!! 仕入管理 KCS-C-1012PJ	商品数 140，仕入先数 100〔プリンター出力〕商品別仕入日計，仕入先別仕入日計表，商品別仕入月計表，仕入先別仕入月計表，仕入先元帳，商品元帳 〔仕入先元帳〕仕入先，住所，TEL，当月買掛金，本日仕入額，本日支払額，当月仕入額，本日現金仕入額，前月請求額，値引，当月支払額 〔商品元帳〕商品名，売単価，月初在庫数，本日仕入数，当月仕入数，本日出庫数，本日出庫金額，当月出庫数，当月出庫金額	¥ 3,000 （HC-20用 ¥ 6,000）	PC8801 FM-11 FM-8 FM-7 X1 HC-20
実戦!! 在庫管理 KCS-C-1013PJ	商品数 140〔プリンター出力〕商品元帳，商品名一覧表，入出庫日報，入出庫月報 〔商品元帳〕仕入管理プログラムと同じ	¥ 3,000 （HC-20用¥6,000）	PC8801，FM-8 FM-7，X1，HC-20
実戦!! 販売管理 KCS-C-1014PJ	商品数 140，得意先数 100〔プリンター出力〕商品元帳，商品一覧表，得意先元帳，得意先一覧表，商品別売上日計表，得意先別売上日計表，商品別売上月計表，得意先別売上月計表 〔得意先元帳〕得意先名，住所，TEL，当月売掛金，本日売上額，本日入金額，当月売上額，本日現金売上額，前月請求額，値引，当月入金額	¥ 3,000	PC8801 FM-11 FM-8 FM-7，X1 HC-20
実戦!! 出庫管理 KCS-C-1017PJ	商品数 140，出庫先数 100〔プリンター出力〕商品元帳，商品名一覧表，出庫先元帳，出庫先一覧表，商品別出庫日計表，出庫先別出庫日計表，商品別出庫月計表，出庫先別出庫月計表 〔出庫先元帳〕出庫先名，住所，TEL，当月不良返品額，本日出庫額，本日入庫額，本日仕上額，当月出庫額，本日不良返品額，前月仕上残額，違算額，当月仕上額 〔商品元帳〕仕入管理プログラムと同じ	¥ 3,000	PC8801 FM-11 FM-8 FM-7 X1
実戦!! 部品展開表 KCS-C-1041P	商品数 200，製品数 8，一製品の部品数45〔プリンター出力〕部品元帳，部品展開表，共通部品表，発注表 〔部品元帳〕部品名，品番，単価 〔製品元帳〕製品名，品番，製品単価，図番，コード，部品名，部品品番，使用数，単価，備考欄	¥ 4,000	PC8801 FM-11 FM-8 FM-7 X1
実戦!! 飲食店売上分析 KCS-C-1051P	テーブル数21，座席数 100，メニュー数 300，1ケ月来客数10,000人以下，1ケ月売上額 1,000万円以下，1ケ月オーダ数10,000以下 〔プリンター出力〕メニュー価格表，テーブル別売上分析表，メニュー別売上分析 〔分析集計内容〕売上数量の入力だけでテーブル別，商品別の売上集計を行い売上比率テーブル回転率，座席回転率，客単価，テーブル単価，メニュー単価，従業員P／H，固定客率の他各種の分析が月中，1日，時間単位にできます	¥ 6,000	HC-20 （PC8801，FM-8 FM-7，X1 用は開発中）
実戦!! 株価分析 KCS-C-1034P	1銘柄単位で 400日分の出来高，株価を管理することによって，星足，移動平均線や逆ウォッチ曲線による分析ができます，特に1画面に移動平均線，出来高，星足を左側に逆ウォッチ曲線を右側に表示し，比較による分析が容易にできるよう工夫されています	¥ 5,000	PC8801 FM-11 FM-8 FM-7，X1
実戦!! 訪問販売 KCS-C-1020P	見込客数60人，見込客の住所，電話番号，見込商品，DM発送日（3回），訪問日（3回）見込度（3回），成約日，成約単価，数量，金額を管理し，DM発送回数，訪問回数，見込度等を指定することによって訪問しょうとする見込客に対してDM用シールの宛名書きや訪問先名等を印刷します，DM発送日，訪問日，見込度は3回を超える分は自動的に書き替えられるので何回までも使用できます	¥ 5,000	PC8801 FM-11 FM-8 FM-7 X1
メールメイト KCS-C-1001P	友達や知人の住所を管理し，必要に応じてハガキやシールに宛名書きができます （シール，ハガキは当社オリジナル，別売）	¥ 3,000	PC8801，FM-11 FM-8，FM-7， X1
デジマンガ KCS-C-1002P	あなたのお好きな絵が簡単にノンプログラムで書け，しかもハガキやストックフォームにハードコピーができ，テレビ画面で塗り絵ができます	¥ 3,000	FM-11，FM-8 FM-7
ピクチャーメール KCS-C-1003P	メールメイトとデジマンガをドッキング，絵の苦手な人や小さなお子様でも簡単に使えます，個性のある暑中見舞，お祝カード，QSLカード等に使えます	¥ 5,000	FM-11 FM-8 FM-7

（※仕入・在庫・販売・出庫管理のいずれかで作った商品元帳は，いずれかのプログラムと共用できます，上記仕様は改良，機種によって異なります）

ザ・パッケージシリーズはさらに新しいソフトを開発中です。その一部をご紹介します。

（PC8801，FM-7，8，11，X1用カセット版）

プログラム名	品番	標準価格	仕様	プログラム名	品番	標準価格	仕様
実戦!! QC管理(1)	KCS-C-1111P	¥ 5,000	特性要因図，パレート図によって品質の管理・改善が行えます。	**実戦!! QC管理(6)**	KCS-C-1116P	¥ 5,000	$\bar{x}$管理図，$\tilde{x}$－R管理によって品質の管理・改善が行えます。
実戦!! QC管理(2)	KCS-C-1112P	¥ 5,000	散布図，ヒストグラフによって品質の管理・改善が行えます。	**実戦!! QC管理(7)**	KCS-C-1117P	¥ 5,000	$\tilde{x}$－R管理図，x－Rs 管理図によって品質の管理・改善が行えます。
実戦!! QC管理(3)	KCS-C-1113P	¥ 5,000	棒グラフ，折れ線グラフによって品質の管理・改善が行えます。	**実戦!! QC管理(8)**	KCS-C-1118P	¥ 5,000	Pn 管理図，P管理図によって品質の管理・改善が行えます。
実戦!! QC管理(4)	KCS-C-1114P	¥ 5,000	円グラフ，レーダチャートによって品質の管理・改善が行えます。	**実戦!! 売掛金管理**	KCS-C-1018PJ	¥ 4,000	入金日計表，売掛金一覧表がプリンター出力ができます。
実戦!! QC管理(5)	KCS-C-1115P	¥ 5,000	帯グラフ，ABC分析グラフによって品質の管理・改善が行えます。	**実戦!! 見積書作成**	KCS-C-1019PJ	¥ 4,000	在庫管理，本プログラムで作成した商品元帳より見積書を作成します。

詳しいお問い合せはお近くのサポートセンターまで……
お買い求めはマイコンショップまたは当社直接でお願いします

近畿コンピュータサービス

福知山サポートセンター
〒620 京都府福知山市厚中町189
春日ビル4F
TEL 0773-22-0419㈹

豊岡サポートセンター
〒668 兵庫県豊岡市正法寺672
TEL 07962-3-5806㈹

高性能パワフル8インチフロッピーディスクユニット

disk-PC M160

新発売　標準タイプ　256,000円

CP/M、MS DOS、P-SYSTEM…………数々のパワフルなSOFTが続々登場してきます。まさにSOFT新時代。

スリムな薄型設計。大容量1MB×2ドライブ。ビジネスに、パーソナルデータベースに。ハイコストパァーフォーマンスを提供いたします。

モデルM160-01(PC9801用)	256,000円
M160-02(PC9801用CP/M-86付)	298,000円
M160-03(PC8801用)	296,000円
M160-05(FM-8用)	298,000円
M160-06(FM-11用)	298,000円

※M160-02以外はシステムディスク(DISK BASIC等)が必要です。接続ケーブル、インターフェイス等はふくまれています。M160-05、M160-06は6月に発売となります。

5"デュアルディスクユニット

快走! disk-80markII

大好評のdisk-80markIIシリーズ。高速に加えさらに静寂・高性能。独走中。

PC8801/PC9801/PC8001markII/PC8001専用

disk-80PII　123,000円

■RACET NEC DOS、CP/M対応 ■PC8001でご利用の場合はdisk-I/OP(12,000円)と接続ケーブルCBL-1(5,000円)をご利用下さい。他はCBL-2(5,000円)でダイレクト接続。システムディスケットは各対応のものをご利用下さい。

FM-7/FM-8専用

disk-80FII　128,000円

■DISK BASICをはじめFLEXなど各DOSに対応。
■FM-7の高速バージョンの対応も可能です。■接続にはdisk-I/OF(15,000円)とCBL-3(5,000円)をご利用下さい。

MZ-80B/MZ-2000専用

disk-80BII　138,000円

■フロッピーインターフェイス内蔵。拡張I/Oポートがあればあとは接続ケーブル(CBL-3)でダイレクトIN。

株式会社 アイテム

〒251 藤沢市南藤沢8-1-202　TEL(0466)27-1668(代)

□お求めは全国マイコンショップへ。
□電話によるご注文もうけたまわっております。
0466-27-1668(AM9:00~PM8:00)
□詳しいお問合せは左記あてへお申込み下さい。
□disk専用クレジットが用意されています。

＊CP/M®はデジタル・リサーチ社の登録商標です。＊UCSD Pascal ™ はカリフォルニア大学理事会の登録商標です。＊FLEX®はTSC社の登録商標です。

北海道帯・広から

FMファンにおくる

PC・MZに負けない 本格的なゲームソフトを!!

(★発売予定……ドドンキ・ゴリラFM、その他 etc)

FM-SL ¥2,800

FM-8、FM-7 マシン語

2月号のI/OでPC用にのっていた、「あのゲームをFM-8、FM-7で」の声にこたえて発売。
汽車ぽっぽを駅に誘導するのが君の使命だ。

MENGO ¥2,800

FM-8、FM-7 マシン語

あの有名なTVゲームの〝PENGO〟を人に変えて、エイリアンを退治する。マシン語だから速い！迷よっていたらすぐやられるぞー。君は何ビキ退治するかな。

THEおろち ¥2,800

FM-8、FM-7 マシン語

マイコンゲームの本4、PC用にのっていた。「あのゲームをFM-8、FM-7で」の声にこたえて発売。
リアルタイムゲーム＋少し頭のゲーム。
とにかく首ののびるおろちを激退せよ。

エーリアンFMパニック ¥2,800

FM-7 マシン語

あの有名なTVゲームスペースパニックがついにFM-7に登場。すごい人気で品薄。
2段落し、3段落しまで出てきて、まだ1人も最後の面までいけないゾ〜。

神経衰弱 ¥2,800

FM-8、FM-7、BASIC＋マシン語

トランプゲームの神経衰弱。
君1人でする時はコンピュータが相手になるヨ。レベルを選んでSTARTだ。
最高4人まで選べるヨ。

七ならべ ¥2,800

FM-8、FM-7、BASIC＋マシン語

トランプゲームの七ならべ。コンピュータが3人分MIE、YUKO、JUNと君の4人でゲームが進みます。勝つためにはジョーカーが決めてになるぞ、BIOS利用によりグラフィックは速い、BIOSの勉強になる。
コンピュータの名前は製作者の？

その他

〈FM—8.7〉ブラックジャック(BASIC＋マシン語)
〈FM—7〉マルパック(マシン語) …………各 ¥2,800

〈MZ-1200〉MENGO(マシン語)・COSMO WARS(マシン語)
ぎゃらめん(マシン語) ……各 ¥2,800

HUSTLE(BASIC)・〈MZ-700〉オセラ(BASIC) …………各 ¥1,500

数々の特典あります。

コンピュータショップ
Byte In

〒080 北海道帯広市西17条南3丁目(競馬場通り) ☎(0155)35-6781

★通信販売を御希望の方は、品名と氏名、住所、電話番号を記入の上、代金＋送料(300円)を、現金書留、又は郵便振替にてお送り下さい。

ビジネスユース、パーソナルユースに合わせて選べる、多彩なソフト パソコンさらに、パワーアップ――。

パソコンの概念を変える数々の機能を、高電社はさまざまなオリジナルソフトで実現してきました。これは、ソフト開発に定評ある当社ならではの快挙。これからも、パソコンをいっそう有効にご活用いただくためビジネスフィールドを中心に、ニーズに応えるソフトウェアをどんどん開発していきます。ご期待ください。

簡易言語シリーズ

簡易言語 SERIES

情報検索型文字データベース

PARAM/K1 漢字

FM-8 ●PC-8800 ●マイブレーン3000……… ¥49,000

1. 項目(データ名)の数と長さ、画面、プリンター出力が自由設定できます。
2. 並べかえ、追加、修正、削除は簡単。
3. 1件(1レコード)64文字から128文字まで。
4. 複合条件(AND、OR、NOT)で検索します。
5. 1行は漢字仕様で53文字。
6. 複合条件(例えば東京都、男性、25才以上、未婚)で必要なデータを検索して表示印刷します。

パソコンは日常語で！プログラミングは自由自在!!

PARAM/1·2·3

パラム1(情報検索型)――――使用機種FM8・PC-8000・PC8800 ¥39,000
パラム2(縦横計算型)――――使用機種FM8・PC-8000・PC8800 ¥39,000
パラム3(マトリックスグラフ作成型)――使用機種FM8・PC-8000・PC8800 ¥39,000

ワードプロセッサー・ソフトシリーズ

マイレター〔日本語ワードプロセッサー〕漢字

FM-8 ●PC-8800 ●マイブレーン3000……… ¥49,000

(カナ漢字変換方式)
1. 使用文字種 漢字JIS第1水準2965文字、非漢字453文字。
2. ディスプレイ表示文字数 40字×20行(10行)。
3. 単語事典20,000語登録 (オプション¥20,000)
4. 訂正・挿入・削除は簡単。
5. アンダーライン、センターリング、可能。
6. 禁則処理有 1文章＝40×100行。

●縦書き、倍角文字可能。

英文ワードプロセッサー〔ワード9000〕

FM-8 ●PC-8800 ●マイブレーン3000……… ¥33,000

医学用ソフト(大阪大学医学部マイコンクラブ監修)

RIA-MATE〔ラジオイムノアッセイデータ処理プログラム〕

FM-8 ●PC-8800 ●PC-9800 ●PC-8000…… ¥70,000

1. 通常の検査室でホルモンの測定等に使われるRIAのCompetitive methodに対しては、logit-log 2次多項式を用いて回帰しており、様々な測定物質に対し良好な回帰が得られる。
2. IRIやAFPの測定及び免疫関係の研究室でもよく使用されるRIAのnon-Competitive method (sandwich method)に対しては、4係数logistic曲線を用いて回帰しており、良好な回帰が得られている。
エンザイムイムノアッセイ(EIA)にも使用可能である。
3. 標準曲線のグラフが得られプリンターに出力可能である。
4. 標準曲線の不適合度を調べて、回帰が良くないときは、飛び離れ点を省いて計算しなおすことが出来る。

ビジネスソフトシリーズ

BUSINESS SOFT

漢字人名簿〔DM宛名書可能〕

FM-8 ●PC-8800 ¥49,000

1. 本システムは名刺整理、顧客資料、会員など多数の人名を登録しておき、必要な資料、宛名等を即座に作成します。
2. 1枚のフロッピーで2200名迄登録、検索、追加、修正、削除、並べかえができ、一覧表や宛名印刷ができます。

カタカナ仕様ビジネスソフトシリーズ

ESCO2000〔見積実行予算システム〕

FM-8 ●PC-8800 ●PC-8000……………… ¥90,000

1. 提出用見積書、原価見積書が同時に作成できます。
2. 資材、工賃、諸経費等の分類・集計を迅速に行います。
3. 実行予算の項目指定は自由、予算の作成は簡単に行います。
4. 実績の消化状想は随時、ワンタッチで見られます。
5. 実績の明細は2000行まで記憶しています。
6. カタカナ仕様

株式会社 高電社
マイクロプロジャパン一次代理店

NECマイコンショップ・システムイン高電社
〒546 大阪市東住吉区杭全7-10-15 ☎(06)719-1131
大阪駅前店
〒530 大阪市北区梅田1-11-4大阪駅前第4ビル6F ☎(06)341-3371
東京営業所
〒101 東京都千代田区神田須田町1-14-6(須田町交差点角)☎(03)256-3061

	カリキュラム	内容
中学一年	＊正の数・負の数(上)(下)	負の数の意味、数直線、絶対値、負の数を含む加減乗除、交換・分配・結合法則、逆数、混合計算など
	一元一次方程式	等式の性質、移項、方程式の解法、(　)・小数・分数を含む方程式、応用問題(速さ、混合問題など)
	比例	比例と$y=ax$、$y=ax$のグラフの書き方、aの値とグラフの関係、グラフ→$y=ax$、反比例、双曲線、比例の応用問題など
	反比例	反比例の意味とその応用、$y=\frac{a}{x}$のグラフ(双曲線)、式とグラフの関係、式の求め方、具体例と応用など
	文字式	積・商の表わし方、式の値、単項式の計算など
	数の計算のすべて	二整数・三整数の加減乗除、分数・(　)・指数を含む計算、加減乗除の混合計算、複雑な計算など
中学二年	＊不等式(上)(下)	不等式の表わし方、符合と不等式、近似値の不等式表現、不等式の解き方、2つの不等式、不等式と数直線、不等式の応用問題など
	一次方程式とグラフ	$ax+by=c$のグラフ、軸に平行な直線の式、連立方程式の解とグラフの関係、練習問題など
	一次関数	$y=ax+b$のグラフ、$y=ax+b$と$y=ax$のグラフの関係、aとbの意味、変化の割合、連立一次方程式のグラフによる解法など
	連立方程式	計算、解法、二元一次、三元一次、応用など
	多項式の加減	多項式×多項式、式の展開、因数分解など
中学三年	＊平方根(上)(下)	平方根の定義、$\sqrt{\ }$の意味、平方因数、$\sqrt{\ }$を含む式の計算、$\sqrt{\ }$の有理化、$\sqrt{\ }$の近似値の求め方など
	関数とグラフ	$y=ax^2$のグラフ、二乗に比例、二次方程式のグラフによる解法など
	二次方程式PART1	二次方程式とは？ 二次方程式の解き方→$ax^2=b$、$(x+a)^2=b$、平方完成、$ax^2+bx+c=0$、応用問題など
	式の展開	(単項式)×(多項式)、(多項式)×(多項式)、公式による展開→$(a+b)^2$、$(a+b)(a-b)$、$(x+a)(x+b)$ 三項式以上の複雑な展開など

価格　ブック式(A) 4,500円（カセットテープ1本・プログラムリスト）
　　　ブック式(B) 6,800円（カセットテープ2本・プログラムリスト）　＊はカセット2本入り

※お求めは京都本部宛に商品名記入の上、現金書留又は郵便為替〔京都6-9053㈱オーク〕宛（送料無料）お申し込み下さい。尚、有名マイコンショップや書店でもお求めいただけます。

発売元 株式会社 オーク

マイコン事業部　〒615 京都市西京区川島有栖川町51 オークビル3F　☎075(391)0391
東京支店　〒102 東京都千代田区四番町三番地 四番町ハイツ303号 ☎03(262)0284

資料請求券 京都本社宛 Oh!FM

作表無限、これは使える。

即実戦力。ビジネス経営に威力を発揮するFMCALCです。

ソフトマート簡易言語アプリケーション

FMCALC

FUJITSU MICRO 7 8用

ディスクバージョン……………………¥68,000
カセットバージョン……………………¥15,000

簡潔な命令で必要十分な処理を！

ワークシート方式で集計・ソート(アイウエオ順等の分類・並べかえ)などのわずらわしい業務を簡単なオペレーションで処理することができるようになります。FM-7、FM-8のハードウェアを効率的に使用することと、できるだけ少ない命令語で目的の処理を行なうことに工夫がこらされています。大企業の各セクションや、中小企業の経営等に威力を発揮する高速・高能力ビジネスソフトです。

主な特長

- ●POWER ONでFMCALCが起動します。他のソフト(FLEX、BASICなど)は不必要です。
- ●たいへん見やすい画面構成になっています。メッセージもできるだけ日本語です。
- ●数値精度は16桁、関数精度は13桁と、高精度の試算を高速で実行します。
- ●FM-7、FM-8のハードウェアと6809CPUの特性をフルに使用した機械語で書かれています。

TSC社日本総代理店・スーパーソフト社 JAPAN AUTHORIZED DISTRIBUTOR

ソフトマート株式会社
〒101 東京都千代田区神田須田町1-18-6 第1谷ビル ☎(03)-256-5881

■取扱店

INTERRUPT東京 ☎(03)256-6325
INTERRUPT横浜 ☎(045)312-2325
INTERRUPT大阪 ☎(06)245-7575
INTERRUPT福岡 ☎(092)671-2466

MICOM CENTER 60

池袋東口に待望のパソコン大型総合サービス店—ますます絶好評！

マイコンセンター60

ハードから
ソフトまで大量品揃え、
技術サポートも万全！

知的好奇心は、
あなたの未来を明かるく
切り拓いてゆきます。

オフコンにせまる超本格派 FM-11

16ビット・8ビット両用パソコン

Aセット

- FM-11EX(MB25050)
- 増設ドライブ(MB27609)
- 専用カラー(MB27311)
- 専用プリンター(MB27403)
- システムソフト(CP／M-86, F-BASIC, 簡易言語FMCALC)
- プリンター用紙1,000枚とディスケット10枚サービス

合計標準価格 799,400円 ➡ ウルトラプライス

クレジット(例)

頭金0円	月々 8,900円×60回	ボーナス 5万×10回
頭金0円	月々12,300円×60回	ボーナス 3万×10回
頭金0円	月々14,800円×60回	ボーナス 1万5千×10回

Bセット

- Aセットの専用カラー(MB27311)のかわりに専用グリーン(MB27312)で他はAセットと同じ

合計標準価格 659,800円 ➡ ウルトラプライス

クレジット(例)

頭金0円	月々 5,900円×60回	ボーナス 5万×10回
頭金0円	月々 8,400円×50回	ボーナス 5万×10回
頭金0円	月々13,000円×36回	ボーナス 5万×6回

先端技術が夢中にさせる興奮パソコン FM-7

- FM-7
- 当社推奨高解カラーRGB-III
- 適正FMデータレコーダー

合計標準価格 259,000円
⬇
ウルトラプライス

クレジット(例)
頭金0円 月々4,200円×36回、
ボーナス 18,000円×6回

今一番ナウイパソコンSHARP

本格派・未来派には
SHARPがピッタリ！

テレビ画像とのクロスオーバー
コンピューターグラフィックス

パソコンテレビX1

- ローズレッド
- メタリックシルバー
- スノーホワイト

見る・創る・
学ぶ・遊ぶ

スーパーインポーズ機能
サウンドジェネレーター
他盛沢山

- CZ-800C/D
- グラフィックRAMCZ-8GR
- ソフト5本サービス

合計標準価格 300,000円 ➡ ウルトラプライス

クレジット(例)

頭金0円	月々 4,400円×60回	㋭ 1万×10回
頭金0円	月々 3,600円×36回	㋭ 3万3千×6回

お電話1本、
お葉書1枚で！
全国即納システム！
(配送料は無料)

あくまで良心的な
営業サービス方針は
熱烈大好評!!
今後の拡充にますます
ご期待下さい。

日本一安いクレジット！
1回から60回までの自由自在。
マイコンの新名所
池袋・東口。

お申込み方法

①商品名(メーカー、型番) ②合計金額(定価合計でも自動値引きされますからご安心ください) ③お支払方法(クレジット、現金、またはリース) ④クレジットの場合…月払い及びボーナス払いご希望額(ボーナス払いを多少なり入れますと月額は低くなります) ⑤お名前(20歳未満の方は保護者のお名前も) ⑥年齢⑦ご住所⑧電話番号⑨職業⑩他、ご意見、ご希望以上を官製葉書にご記入の上、右記宛ご郵送ください。

※お金のやりとりは1円なしでも、お手元に全品が揃います。(頭金0円のとき)

●銀行振込ご利用の方は住友信託銀行池袋支店・普通口座No.2706052にお振込みください。[口座名:(株)マイコンセンター　送金手数料は差引きもOK]

お問い合せ・お申し込みは

マイコンセンター60
Oh／FM係
☎03-980-1360㈹
〒170 東京都豊島区東池袋1-21-5
サンシャインシティー出入口前

EDITOR'S ROOM

◆FMのハード・ソフトに詳しい人で，原稿の書ける方を募集しています。簡単な自己紹介，マイコン歴，得意な分野，実力評価用の作品(ハードまたはソフト)など，原稿依頼の資料となるものを郵送するか，電話のうえ直接ご持参ください。

〒102 千代田区四番町2-1 ㈱日本ソフトバンク
Oh! FM編集室 松岡 ☎03(261)4095

◆オリジナルプログラムも下記の要領で募集しています。

- ①プログラムの内容，特徴，アルゴリズムなどの説明，②フローチャート・変数表，③遊び方や操作方法，注意事項 ④テープないしはディスケット
  以上と，住所・氏名・年齢・職業・電話番号などを明記のうえ，郵送するか直接お持ちください。あて先は上記と同様です。
- プログラムは未発表のものに限ります。
- 応募原稿は原則としてお返し致しません。
- 掲載にあたって多少修正させていただくことがあります。
- 掲載の場合は，本誌規定の原稿料をお支払い致します。

◆本誌の内容に関して，電話による質問を受け付けます。

☎03(265)5789 受付時間 16:00～18:00

質問はバグ情報についてのみ受け付けます。機械の取り扱いなどについてのお問い合わせはご遠慮ください。マニュアルを熟読してください。

●広告索引●

- **編集** 渡辺妙子 松岡真理
- **技術** 松田辰夫 小林初雄 徳永 聡
- **海外協力** HONG・LIANG・LU IAN・ALLEN RONALD・BILLINGS HISAYOSHI・MIKAMI
- **協力スタッフ** 木下淳博 桑原岳夫 鶴岡哲明 長沼孝仁 西村義孝 本郷誠司 林 剛正
- **写真** 浜崎 昭 杉山和美
- **レイアウト** 創美レイアウト レモンデザインルーム㈱

編集後記

■本誌の専任スタッフは女性のみ，というのをご存知ですか。中で唯一の既婚女性の仕事ぶりがすさまじい。底冷えのする半地下の事務所で深夜労働を強いられているのです。連日ですよ。ライター諸氏，早く原稿を入れて下さいョ，彼女の健康のために，それにもまして家庭崩壊を防ぐために。(T)

■本誌のスタッフは女性ふたり。ひとりは丸く，ひとりは細いという対照の妙。若い執筆陣に姐御のように慕われているのがチーフの丸いMさん。お尻だけは発達している細いTさん。共通点は美人(?)で有能なところか。本誌の評判がよいのも，この意気の合ったコンビに負うところが大きいようだ。(O)

■本社のあった場所に編集室が引っ越しました。編集室もマシン室もなんと3倍になったのです。FM編集部は最良の場所を確保し，マシンも棚から引っ張り出さずとも，それぞれ机の上にセットされています。これぞデスクトップなのです。……1週間たちました。電源コードがない，机の回りは物だらけ。なぜだ!(M)

■某大手マイコン出版社から明るそうなマイコン雑誌が出版されています。しかし，今一しっくりこない。どうあがいてもマイコンは暗いもので，それを無理に演出しているとしか見えないのです。システムハウスや出版社で働いている人間の何と暗いことか。としか見えない僕は，自分でも暗いと思います。皆さんは? ☺

■今まではたいていの会社と同じく9時出勤だったのですが，このたび10時出勤が認められました。朝の1時間の違いというのは大きいものです。時差通勤実施者だった(過去形)私ですが，10時出勤のおかげで大きな顔ができるようになりました。なにはともあれめでたい，めでたい。(T)

■さあ新学期，社会人になった方，今年こそ，パソコンを使おうという時，まず敵を知る。これ兵法なり。FMのマニュアルは入門書，解説書両方くわしく書かれている。読んで解らない時は，友人や先輩などに相談する。身近にいない人は，shopの店員さんと仲良くして，色々ノウハウを手に入れる。聞くは一時の恥。(H)

隔月刊 Oh!FM 第3号

- 昭和58年6月1日 発行
- 発行人／孫 正義
- 編集人／田鎖洋治郎
- 発行元／株式会社日本ソフトバンク
  本社／〒102 東京都千代田区九段南2-3-14 靖国九段南ビル2F
  ☎03(263)3690 Fax.03(263)3660
  編集室／〒102 東京都千代田区四番町2-1 ☎03(261)4095
  大阪営業所／〒542 大阪市南区難波千日前5-19 河原センタービル
  ☎06(644)0191 Fax.06(644)0160
- 印刷 図書印刷㈱ ☎03(453)2550

本格的ビジネスユースからホビーユースまで FMシリーズ

FM-8

新発売 FM-11

新発売 FM-7

富士通興業の充実したビジネスソフトウェアが FMシリーズを一段と使いやすくします。

究極のパソコンワープロ

新バージョン FM-WORP ver2.0　定価：¥60,000

発売以来使い易さで好評の「FM-WORP」がバージョンアップされました。パソコンでは初めての文節単位の文章入力等一段と使い易さにみがきがかかりました。「FM-WORP」「FM-WORP」は親切設計、見て、さわってお選び下さい。

- 文節単位の文章入力、しかも辞書の学習機能付
- B4/A自由自在、あなたのプリンタに合せて任意の書式が設定できます。しかも自動ケイ線機能付
- 文章名など、会話は全て日本語、入力項目は標準値採用など初心者でも楽々操作できます。

《旧バージョンをお持ちのお客様へ》

「FM-WORP」の旧バージョンをお持ちのお客様で新バージョンへのレベルアップを希望される方は旧バージョン一式に手数料5,000円を添えて当社までご送付下さい。折返し新バージョンを返送させていただきます。

新発売 三次元グラフィックス PERS-F1　定価：¥80,000

コンピュータグラフィックスの中で、パソコンでパースを描かせるという目的で開発されたグラフィックソフトです。画面にドットで線や箱を作る"LINE"、指定座標を直線で結ぶ"CONNECT"、円や円弧を作る"CIRCLE"などの命令があり複雑なビジネスグラフなどを作ることができます。

応用範囲：日照日影図・POP・オフィスビル店舗レイアウト・展覧会・レタリング・イラスト・図案・その他

FMシリーズを最も良く知っているインストラクターが指導するFM-LABO。詳しい案内パンフレットができておりますので、下記のマイコン営業部03-567-3468までどうぞ。

大阪 OA PLAZA 開催中

大阪地区パーソナル・コンピュータスクール開催中です。
会場：大阪市北区堂島1-5-17 堂島グランドビル5F
富士通興株式会社 OA PLAZA TEL06-343-2626
日程：コースなど詳細については、電話でお問い合わせください。

躍進する富士通グループ

富士通興業

富士通興業株式会社

FMシリーズのお問い合わせは……

OA機器営業本部
〒106 東京都港区六本木4-1-4黒崎ビル ☎586-1511

パソコン営業部
〒104 東京都中央区銀座2-6-1中央銀座ビル☎567-3468

札幌営業所 ☎011-221-8501
仙台電子営業所 ☎0222-62-5252
北関東電子営業所 ☎0486-41-1747
名古屋営業所 ☎052-211-5866
大阪電子営業所 ☎06-343-2626
広島営業所 ☎0822-22-6141
九州電子営業所 ☎092-472-4111
九州電子営業所 ☎0963-55-3166

Oh!FM 昭和58年6月1日発行 第1巻第3号通巻第3号

Oh!FM ㈱日本ソフトバンク発行 Printed in Japan 定価480円 雑誌02199-06